'68' MICRO JOURNAL

$2.50 USA

Australia	A $ 4.00	New Zealand	NZ $ 4.00
Singapore	S $ 8.00	Hong Kong	H $20.00
Malaysia	M $ 8.00		

VOLUME II ISSUE 11 • Devoted to the 68XX User • November 1980

"Small Computers Doing Big Things."

SERVING THE 68XX USER WORLDWIDE

PHOTO CREDIT: NASA

UniFLEX™

Multi-User

UniFLEX is the first full capability multi-user operating system available for microprocessors. Designed for the 6809 and 68000, it offers its users a very friendly computing environment. After a user 'logs-in' with his user name and password, any of the system programs may be run at will. One user may run the text editor while another runs BASIC and still another runs the C compiler. Each user operates in his own system environment, unaware of other user activity. The total number of users is only restricted by the resources and efficiency of the hardware in use.

Multi-Tasking

UniFLEX is a true multi-tasking operating system. Not only may several users run different programs, but one user may run several programs at a time. For example, a compilation of one file could be initiated while simultaneously making changes to another file using the text editor. New tasks are generated in the system by the 'fork' operation. Tasks may be run in the background or 'locked' in main memory to assist critical response times. Inter-task communication is also supported through the 'pipe' mechanism.

Support

The design of UniFLEX, with its hierarchical file system and device independent I/O, allows the creation of a variety of complex support programs. There is currently a wide variety of software available and under development. Included in this list is a Text Processing System for word processing functions, BASIC interpreter and precompiler for general programming and educational use, native C and Pascal compilers for more advanced programming, sort/merge for business applications, and a variety of debug packages. The standard system includes a text editor, assembler, and about forty utility programs. UniFLEX for 6809 is sold with a single CPU license and one years maintenance for $450.00. Additional yearly maintenance is available for $100.00. OEM licenses are also available.

FLEX™

UniFLEX is offered for the advanced microprocessor systems. FLEX, the industry standard for 6800 and 6809 systems, is offered for smaller, single user systems. A full line of FLEX support software and OEM licenses are also available.

technical systems consultants, inc.

Box 2570, West Lafayette, IN 47906
(317) 463-2502 Telex 276143

'68' MICRO JOURNAL

Portions of text prepared using the following.

SWTPC 6800-6809-DMAF2-CDS1-CT82-Sprint 3
Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216

EDITOR - WORD PROCESSOR
Technical Systems Consultants, Inc.
Box 2573, W. Lafayette, IN 47906
FLEX is TM of TSC

GIMIX Super Mainframe-Assorted memory boards
GIMIX Inc.
1337 West 37th Place
Chicago, Il 60609

Publisher: Don Williams Sr.

Executive Editor: Larry Williams

Subscriptions and Office manager
Mary Robertson

General Girl 'Friday'
Joyce Williams

Contributing Editors:

Dr. Jack Bryant
Dr. Chuck Adams
Dr. Theo Elbert
Dr. Jeffery Brownstein
Dale Puckett
Russell Gore
Ron Anderson
John Jordon

* CONTENTS *

Send All Correspondence To:

'68' Micro Journal
3018 Hamill Rd.
PO Box 849
Hixson, Tennessee 37343

— Phone —
Office: 615-870-1993
Plant: 615-892-7544

'68' Micro Journal is published 12 times a year by '68' Micro Journal, 6131 Airways Blvd., Chattanooga, TN 37421. Second Class postage paid at Chattanooga, TN. Postmaster: Send Form 3579 to '68' Micro Journal, PO Box 849, Hixson, TN 37343.

1-Year $18.50 2-Year $32.50 3-Year $48.50

—ITEMS SUBMITTED FOR PUBLICATION—

(Letters to the Editor for Publication) All 'letters to the Editor' should be substantiated by facts. Opinions should be indicated as such. All letters must be signed. We are interested in receiving letters that will benefit or alert our readers. Praise as well as gripes is always good subject matter. Your name may be withheld upon request. If you have had a good experience with a 6800 vendor please put it in a letter. If the experience was bad put that in a letter also. Remember, if you tell us who they are then it is only fair that your name 'not' be withheld. This means that all letters published, of a critical nature, cannot have a name withheld. We will attempt to publish 'verbatim' letters that are composed using 'good taste.' We reserve the right to define (for '68' Micro) what constitutes 'good taste.'

(Articles and items submitted for publication) Please, always include your full name, address, and telephone number. Date and number all sheets. TYPE them if you can, poorly handwritten copy is sometimes the difference between go, no-go. All items should be on 8X11 inch, white paper. Most all art work will be reproduced photographically, this includes all listings, diagrams and other non-text material. All typewritten copy should be done with a NEW RIBBON. All hand drawn art should be black on white paper. Please no hand written code items over 50 bytes. Neatly typed copy will be directly reproduced. Column width should be 3¼ inches.

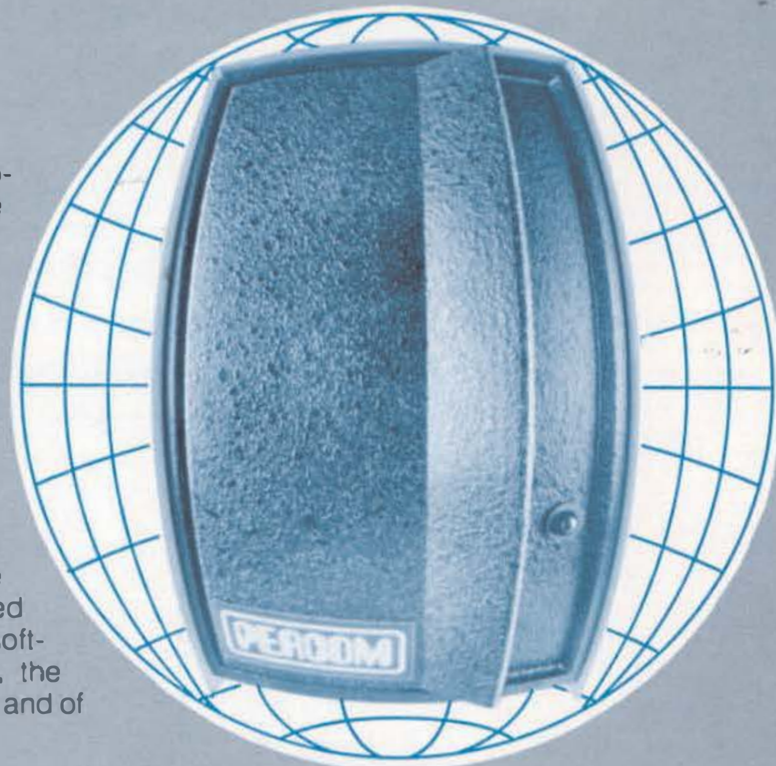
(Advertising) Any Classified: Maximum 20 words. All single letters and/or numbers will be considered one (1) word. No Commercial or Business Type Classified advertising. Classified ads will be published in our standard format. Classified ads $7.50 one time run, paid in advance.

Commercial and/or Business advertisers please write or phone for current rate sheet and publication lag time.

SEE GHOST AD PAGES 44, 47, 52, 56

In the world of 6800 Microcomputing there is only one Universal Mini-Disk System . . .

the PERCOM LFD-400™ with SOFTRAN™



Made possible by SOFTRAN™, an innovative $24.95 translator program, the reliable Percom LFD-400™ has just been upgraded to the first universal mini-disk storage system.

Suddenly the two worlds of 6800 minidiskette software become one. Because the LFD-400™ with SOFTRAN™ can read either soft-sectored or hard-sectored disks.

And owning an LFD-400/SOFTRAN system means you can run minidiskette programs from the enormous combined selection of all of the principal 6800 software houses — TSC, Computerware, the Software Works, Hemenway Associates and of course Percom.

Available in versions for mini FLEX†, FLEX 2.0† and Smoke Signal Broadcasting Company's DOS, SOFTRAN™ copies soft-sectored minidiskettes track-for-track onto hard-sectored minidiskettes. If the source disk includes a FLEX† or 'Smoke' DOS, SOFTRAN™ is used to modify the operating system to function with the Percom LFD-400™.

SOFTRAN™ is supplied on a minidiskette along with utilities for only $24.95. A users manual is included. You must indicate whether SOFTRAN™ is to be used for mini FLEX†, FLEX 2.0† or Smoke's DOS.

The Percom LFD-400™ mini-disk system sells for only $599.95, complete with: (1) the drive, drive electronics and Percom's rugged PS-401 power supply all in a finished enclosure, (2) a demonstrably superior controller PC card featuring an explicit data/ clock separation circuit, MPX, a remarkable 2K DOS, and provision for 1K extra PROM, (3) an interconnecting cable and (4) a 70-page users manual.

Also available: Upgrade kits for SWTP or 'Smoke' mini-disk drive systems. Kit includes LFD-400™ controller, MPX DOS & SOFTRAN™. Only $224.95.

Available soon!
SOFTRAN™ for Percom's 77-track LFD-800™ mini-disk system; SOFTRAN/9™ for 6809 FLEX† files and programs.

™ trademark of Percom Data Company, Inc.

† trademark of Technical Systems Consultants, Inc.

PERCOM

PERCOM DATA COMPANY, INC.
211 N. KIRBY GARLAND, TEXAS 75042
(214) 272-3421

Percom *'peripherals for personal computing'*

To place an order or request additional literature call toll-free 1-800-527-1592. For technical information call (214) 272-3421. Orders may be paid by check, money order, COD or charged to a VISA or Master Charge account. Texas residents must add 5% sales tax.

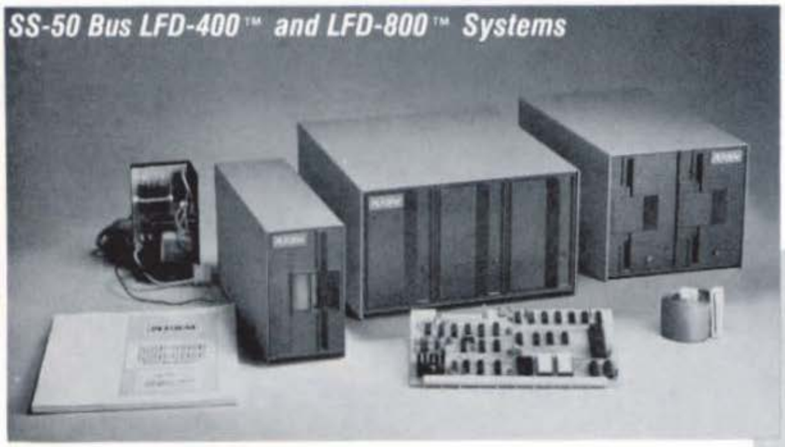
PRICES AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE

A Few Extraordinary Products for Your 6800/6809 Computer

From Percom . . .

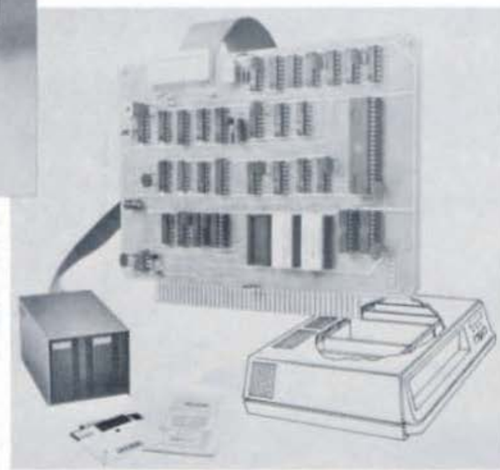
Low Cost Mini-Disk Storage in the Size You Want

SS-50 Bus LFD-400™ and LFD-800™ Systems



Percom mini-disk systems start as low as $599.95, ready to plug in and run. You can't get better quality or a broader selection of disk software from any other microcomputer disk system manufacturer — at any price!

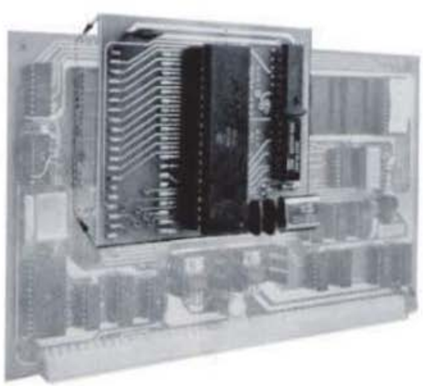
Features: 1-, 2- and 3-drive systems in 40- and 77-track versions store 102K- to 591K-bytes of random access data on-line • controllers include explicit clock/data separation circuit, motor inactivity time-out circuit, buffered control lines and other mature design concepts • ROM DOS included with SS-50 bus version — optional DOSs for EXORciser* bus • extra PROM sockets on-board • EXORciser* bus version has 1K-byte RAM • supported by extended disk operating systems; assemblers and other program development/debugging aids; BASIC, FORTRAN, Pascal and SPL/M languages; and, business application programs.



EXORciser Bus LFD-400EX™-800EX™ Systems*

Full Feature Prototyping PC Boards

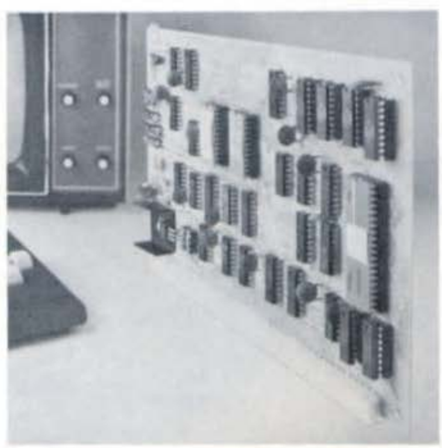
Percom SS-50 and I/O bus prototyping cards include all of the features needed for easy, straightforward prototyping. Use wire wrap, wiring pencil or solder wiring. **Features**: tin-lead plating over 2 oz. copper wets quickly, solders easily • provision for power regulators and distributed capacitor bypassing • SS-50 bus card accommodates 34- and 50-pin ribbon connectors on top edge, 10-pin Molex connector on side edge — costs only $24.95. • I/O bus card is 1-1/4" higher than SWTP I/O card, accommodates 34-pin ribbon connector and 12-pin Molex connector on top edge — costs only $14.95 • Both card designs accept 14-, 16-, 24- and 40-pin DIP sockets.



This 6809 upgrade adapter may be used on the SWTP 6800 and most other 6800/6802 MPU cards. Supplied assembled and tested, it costs only $69.95 with user instructions. The original system may be restored by merely unplugging the adapter and a wire-jumpered DIP header, and re-inserting the original components. Also available for your upgrade computer is PSYMON™, the Percom SYstem MONitor for the Percom 6809 single-board computer. PSYMON™ on 2716 ROM costs only $69.95 — PSYMON™ is also available on minidiskette, with source and object files, from the Percom Users Group.

The Electric Window™: Instant, Real-Time Video Display Control

This VDC card resides completely in main memory so that control is accomplished instantly by direct MPU access to the on-board 2K character-store memory and the display control registers. Price is only $249.95. **Features**: Programmable CRT controller chip provides extraordinary capability for software control of functions such as number of characters per line, number of lines displayed, highlighting and interlaced or non-interlaced scan • includes ASCII 128-unit character generator which generates 7-dot by 12-dot characters — lower case letters have descenders • provision for optional ROM for special characters/symbols • comprehensive manual includes full listing of WINDEX™, the Electric Window™ driver program — WINDEX™ is also available on minidiskette through the Percom Users Group.



™ trademark of Percom Data Company, Inc.
* trademark of the Motorola Corporation.
Prices and specifications subject to change without notice.

PERCOM

PERCOM DATA COMPANY, INC.
211 N. KIRBY GARLAND, TEXAS 75042
(214) 272-3421

Products are available at Percom dealers nationwide. Call toll-free, 1-800-527-1592, for the address of your nearest dealer, or to order direct.

HEAR YE!

HEAR YE! HEAR YE!

SOFTWARE ANNOUNCEMENT

NOW INCLUDES LIBJCP

JCP
Job Control Program
By Peter Murray

The JOB CONTROL PROGRAM (JCP) reads a text file that contains the necessary input for a program and then supplies this input to the program in the same manner that an operator would have normally entered it from the keyboard. The file containing the input is referred to as the procedure file, and the program receiving the input is referred to as the calling program. A procedure file contains input for such calling programs as FLEX™, FLEX utility commands, and other development software.

LIBJCP is used as a FLEX™ command within a procedure to load and execute another procedure. At the conclusion of the called procedure, control will return to the calling procedure and execution will resume at the line following the LIBJCP command.

JCP also provides for parameter substitution within the procedure file, special commands to control JCP program flow, and a means for recovery from processing errors. These features allow for commonly used file routines to be written as a generalized procedure that JCP will execute, unattended, simply by entering a single FLEX command.

PRICE $89.95

See Review in July '80 '68' Micro.

AVAILABLE NOW!

REMOTE
Intelligent Terminal Program
By Tom Speer

REMOTE allows use of your 6800 or 6809 system as a terminal to a remote computer. REMOTE gives you access to the new, inexpensive time-sharing systems designed for home computer users, such as MICRONET or THE SOURCE, or any other time-sharing system. You can also use REMOTE to talk to another microcomputer system. All you need is your FLEX based 6800 or 6809 disk system, a serial interface, a modem, and REMOTE.
REMOTE is flexible. You can control its many features and options thru local commands at any time. Output from the host system (from the phone line) may be directed to the console CRT, the system printer, or a disk file, or any combination of the three. You can change parity checking, control character definitions, etc. You can read files from disk to the phone line under user control. You also can issue any FLEX command while you are not receiving data.

INTRO PRICE $39.95

REMOTE is a terminal program designed to give you the utmost in flexibility and convenience with as many different timesharing systems as possible. It was written by Tom Speer, P.E., an Electrical Engineer with over twelve years of data communications hardware and software experience.
Remote will support the New Thomas Instrements Modem Card.

NEW

READTEST
English Text Analysis Program
By Dale Puckett

READTEST is a must for all writers and writing instructors. Reads prose from disk file and tells how well it was written. Reports number of lines, words, sentences, personal words, affixes, average sentence length. Individual reports pinpoint trouble areas. Overall index tells who can read it and who would print it. Fast 6800 object code. Runs in FLEX™. READTEST, a 6800 assembly language program written by 68 Micro Journal Contributing Editor Dale L. Puckett, is such an indispensable applications program for all writers, fiction or non-fiction, student or professional. READTEST reads your letters, short stories, news stories, reviews, novels, etc., from the disk file you prepared them on and tells you how well they were written. READTEST is a powerful tool designed to aid both the student writer and keep a running check on the experienced writer. It is a serious application of your 6800 microcomputer that will pay for itself with the first check you receive for a readable manuscript.

APPROX. 12K : INTRO PRICE $39.95

ESTHER
An exercise in artificial intelligence
By Dale Puckett

ESTHER is Eliza plus. Artificial intelligence in pure 6800 code. Her source shows you how. Her object will amaze your friends. ESTHER: remembers names, drops them, uses the player's name, and even echos keywords. ESTHER identifies more than 75 keywords and uses almost fifty sets of replies. A few of the sets contain as many as 21 replies to help her avoid redundancy. ESTHER features auto line length and runs in FLEX™. She obeys TTYSET. She is both educational and fun. ESTHER, written by 68 Micro Journal Contributing Editor, Dale L. Puckett, is the result of a two year long experiment with artificial intelligence in 6800 assembly language programming. ESTHER randomly inserts the players name in the conversation. Occasionally, she uses part of the player's reply in the middle of her answer or next question. ESTHER has the ability to echo keywords. This allows her to respond to replies from the player which are in the third person.

ESTHER identifies proper nouns and uses them in her replies. She also saves them for later use.

APPROX. 16K : INTRO PRICE $39.95

FLEX is a trademark of Technical Systems Consultants, Inc.

master charge VISA

FRANK HOGG

DENTAL LABORATORY
130 Midtown Plaza
700 East Water St.
Syracuse, N.Y. 13210
(315) 474-7856

All software is currently available on FLEX™ 2.05* disks soft sectored disks. The package includes a users manual, the disk with object code AND FULLY COMMENTED SOURCE LISTING, a programming anual with information about the program and hints for changes, and where applicable example programs.

VISA and MC accepted. SOURCE TCF339.

Now available DMAF 8" Disks.

ATTENTION PROGRAMMERS!
We are looking for quality software to market. Contact Frank Hogg.
*6809 software should be ready by the time this ad is out.
Call for availability.

PHILLY SHOW

A grouping of FIFTY BUS exhibitors was organized into one section at the recent Personal Computing show held in late August in Philadelphia. This section was only for 68xx products, and was the largest area in the show. In addition to the exhibit booths there was a combination meeting and lecture area set up in the center. Speakers included Tom Hensen of Computer Consultants of Michiana, Dave Shirk and Den Vanada of TSC, Bob Bundy of Sonex, Ray Talbot of Kenyon, John Alford, Joel Heckman of Universal Data Research, Ken Kaplan of Microware, and Shel and Sam Epstein, with Don Williams serving as ringmaster.

There were 15 separate exhibit booths surrounding the meeting area in a town square type of setup. A synopsis of each booth follows.

Thomas Instrumentation highlighted Mark Sproul's fully automatic answer/originate Modem board. They also showed their new low cost video based 6800 system. Jan Gluyas caught your reporter sneeking Tom 2 butterscotch candy balls. He gave back 4. For those of you who have spoken to Donna on the phone, she's even cuter in person.

Sirius Systems was staffed by Jerry and Leah Robinson who will be having a Forth and a disk controller card for 6809 systems.

CSI's UCSD Pascal was shown by Dave Allen and Kathy Micken.

Universal Data Research demonstrated their Date Base Management System. Joel Heckman has put a lot of time and effort into the developement of this, but give Joanne a lot of credit for her support to Joel.

Sonex System's Stylograph caused an editor of a computer period ical, who shall remain unnamed, to exclaim "I didn't know the SS50 had software." He later

explained that he meant good applications software, but admitted that his publication did not cover or solicit software reviews for our bus. He has promised to correct this in the future. This was the first show for Bob Bundy and the first chance for us to meet him in person. He's a quality guy, as is his software.

Kenyon Microsystem's demonstrated their Forth. Ray Talbot was one of the original developers of Forth, and his credentials and background are impressive, but in spite of that he is a real regular person. And a special thanks to Tom Kenyon who hosted a nightly cocktail hour in his hotel room where we could meet and decide where 20 some odd people could feed at one table. The job of Restaurant reviewer has been given to one Don Williams, so look for his column giving tasteful or tasteless suggestions on eating out in Philadelphia.

GIMIX' booth was staffed by Mike Magnus and Richard Don. They showed for the first time their new Disk Controller cards, as well as demonstrating GIMIX' version of TSC's Flex and Microware's OS-9 in use on their 6809 systems. Another GIMIX system was dedicated to showing their 512x512 high resolution graphics, which proved to be a showstopper. The other new GIMIX cards at this show were the 16 socket EPROM/ROM/RAM board capable of storing up to 128K bytes, and their 2 port and 8 port Serial I.O. boards.

The Microware booth was always busy. Ken Kaplan and Larry Crane had a field day demonstrating and teaching users how to use OS-9 and Basic 09. The sales and pending license agreements that resulted from the show had them beaming as they had proven that they have a winner. (note to editor-please identify Larry's photo. In previous story on S.F. Faire he was the unidentified faire goer. At this show he had a badge identifying himself as such) (OUCH! ye ole ed). Ken's

Sunday afternoon symposium in our meeting area drew the largest crowd, we had.

Another first time exhibitor was Star-Kits. Pete Stark of course is no stranger to users on the FIFTY BUS. This was his turn however to be on the other side of the table. Altho he expected the most interest to be in his software, it was his SBC-02 single board computer that stopped the lookers. At this point let me exercise my writer's privilege to digress into Pete's after show hours activities. First, thanks for sharing my double chocolate cake with chocolate ice cream. (note to our wives-we're only kidding) (Ed's note; NO he is not!). Thanks for bringing your friend, Joe Melhado, whose stories and jokes can not be printed, but kept us drunk with laughter into the wee hours. Now if Don will only run that picture of Pete you readers will get a real insight into what Pete Stark is really like (Ed's note; it's here).

The next booth we visit was staffed by Joyce Williams with some help from Don. You should have seen that gal sell old copies of 68 Micro Journal. It seems that once a new subscriber starts, he wants all the back issues. Joyce was there to open up with most of the rest of us bright and early each day. Not so with Don. He kept the rest of us up leading the bull sessions in the Hilton Lobby until early A.M. And then he slept in the morning while Joyce and the rest of us worked pie-eyed. Next year I'm going to hire Joyce away from him and get even. The real hero on Saturday was Ken Kaplan. He and Don were up until 5:30, and Ken was at the booth at 8:30. (Sunday was another matter tho, wasn't it Ken?) Also helping out at the 68 Micro booth was an old friend from past shows who enjoys himself so much that it rubs off on all of us, Joe Sobieski. One of the great enjoyments one gets from participating in these shows

is the people you meet and get to know. Our industry is in its youth, and it and the people involved in it have the vitality of youth and I know of no greater example then Joe (Ed's note; agreed 100%).

Hazelwood Computer Systems is another first timer. Dave Bridger showed (and sold the first day all the boards he brought) their new 256x256 graphics card. He and Mike Smith are winners. We have convinced them to go ahead with an IEEE 488 interface card which we feel is needed for the bus and which is still in the preliminary design stages.

Frank Hogg cut his first set of teeth at this show. They demonstrated and answered questions on various new software written by himself, Pete Murray, Tom Speer, and Dale Puckett, none of who should be

strangers to readers of this journal. By the way, in case Don forgets to give photo credits, they belong to Dale Puckett and Mike Magnus.

The febe group showed the prototype of their FIFTY BUS mainframe with built in keyboard. It looks like a viable product and was well received by the viewers. Watch this journal for announcement of when it will be available. If the production models are equal to the proto type, I predict they will receive a very favorable review.

John Alford is another familiar face who for the first time was on the other side of the table. He received a lot of interest in his CONFORM and SCREDITOR packages. He is also a proud father and rightly so.

We end our tour with HHH. Tom Harmon and Dora Horne demonstrated their SPIRIT language and their ability to put it all together. They had a real nice color graphics demonstration. One of my personal highlights was when I introduced Bob Lentz of Microworks to Tom as Curt from Germany. Bob and Tom had been old friends over the phone, but had never met in person. But when Bob came out with "Vos is los?", Tom cracked up and knew he was being had. To Dora went the honor of being "Miss NIFTY FIFTY."

A special thanks to John Dilks, the show's promoter, who cooperated in helping us have our own special section. He and Shery and Dave and Janis Jones arranged for signs, seating, podium, amplifiers, and other amenities that allowed us to create our own convention within the show.

EAT YOUR HEARTS OUT. Don Williams is jealous. Pete Stark gives a victory salute as Richard Don introduces Miss NIFTY FIFTY, Dora Horne.

The show is also people who attend. I personally met with dealers from England, France and Canada as well as the U.S.. Universities from Canada and the U.S. were represented. Other industrial users of FIFTY BUS products were in the fields of Aerospace, Nuclear Research, Military, Auto, Tire, Electrical Equipment, Communications, Security Systems to name a few who come to mind as I write this. I dare not list names for fear of leaving someone out.

We, as a group, created the impression among all the visitors to the show that the FIFTY BUS had the most to offer and the most advanced products. Our unified area was not only the largest, but had more variety and

Ken Kaplan of Microware addresses the largest turnout, for speakers. His subject was OS-9 and BASIC09.

Waiting for his turn on the podium is Dan Vanada of TSC. That's Matt Scudieri of Oak Ridge on the right.

more new products, both hardware and software, than any other bus. We generated among ourselves a new espirit-de-corps. At our Thursday night-Friday early morn lobby bull session one of our users was describing the accuracy and tolerances he needs and gets from his 160K 6809 system. The system is being used in the most advanced space age technology. The others present shared our pride and had their eyes opened to the potentials we have in the future. We have come a long way in a short time, but still have a long road ahead. We hope to have this user write an article in the future.

This show proved to exhibitors, visitors, and press what we can do when we present a unified picture. For the exhibitors it was profitable. The users got to see, use and question and learn about the latest in hardware and software in the most convenient manner. To the press and others not on our bus, it was an impressive job of public relations. We shall do it again next year.

Richard Don

Ed's Note: Due to space we will run some of the pictures indicated above in December's issue.

DMW

Rumors?

The following is gathered from multiple sources and the accuracy cannot be verified for every item referenced. However, there should be more complete announcements to follow in future issues of 68 Micro Journal.

From Southwest Technical Products are some new CRT terminals and additional 5" disk systems. Some of the new 5" systems store more data than earlier 8" sytems. Also nearing completion is a new text editor and output formatter (this is done on the formatter (preliminary version)). Uniflex for the SWTPC S09 systems is now being shipped. Also a new version of 09 FLEX™ is being shipped by SWTPC and includes a host of new and useful utilities.

GIMIX is now shipping their new 6809 systems and CPU cards. Also in the next month or so they promise to begin shipping their complete line of disk controller cards, both 5 and 8 inch versions. Included is a 'DMA' 8 inch disk controller board, that will work with existing 8 inch disk systems.

We do not, as of this writing, have either the 8 inch version of OS9™ from Microware or Uniflex™ from TSC running. We hope to have them both running by next month and maybe I can let you know a little about each.

Look for a new 6809 computer from the ATARI folks. Look for 32K expansion kits, disks, printer and other products to be advertised soon. I feel that many of our S50 bus vendors should have a field day with this machine. No firm info but the rumors are fairly straight.

Published in this issue is as much technical data as we can scrounge up on the SAM MC6883 Synchronous Address Multiplexer chip produced by Motorola for the new Radio Shack TRS80C color computer, using a 6809E. This chip (6883) is a new device and has little (from reports gathered) field testing. Reports so far is that it runs 'hot' in the TRS80C and could possibly be a source of trouble. Later versions will take care of this problem.

From down Texas way (Digital Research Computers) come information of three new products for the S50 bus. A single board 6809 computer, I/O, 64K RAM (with provisions for extended addressing to 1 Megabyte), color graphics 'DAT' type monitor and other facilities including a video generator built in, rumored price $795.00. A 64K dynamic board in kit, with sockets, to sell for less than $300.00, with extended addressing. A 32K static, 2 Mhz, memory system for around $400.00.

You 68000 buffs need to look into Hemenway's new 68000 O/S. It is reported to be DEC™ type in operation, but faster and with greater utility. Motorola seems to have wrapped up their deal with NCR and IBM for the special 'micro-coded' versions of the 68000.

All you folks over there around Phoenix that heard a loud rumble were not hearing buffalo stampeding; nope, it seems that some heads have 'rolled' because of the condition of the 68000 project. Seems now that there are some bad internal problems, both with the chip and the departments. Where are the suport devices - for that matter where is the 'final' version (that works) of the 68000? We have not seen it yet. One thing for sure, when it actually gets here, it should be a winner, when it get here! More as I hear it.

Bear in mind that much of the above is unseen and as stated, rumors. However, I try to confirm as much as I can. Remember it was over a year ago that we told you about the Radio Shack 6809 unit, which is now being delivered. Some other editors and publishers were saying we were 'all wet'. Now we know!

Dixie

DIXIE is a new 'Disk Executive' for the Percom LFD-400 disk system just introduced by Blue Hat Software Company, Box 4127, Flint Mich. 48504. It differs from Percom as well as other 68xx disk operating systems in many ways and provides many new and useful functions. It provides dynamic allocation of disk space, multiple catalogs which are used to group together different files, 12-character file names, file protection facility, and space for 14 catalogs and 45 files. Its author, Larry Preston, is chief systems analyst at a large commercial Honeywell computer installation. It is easy to see that parts of DIXIE reflect his big-system outlook.

The Basics

Most of DIXIE is on a 2708 EPROM which plugs into the middle EPROM socket on the LFD-400 disk controller, replacing Percom's MINIDOS-PLUSX. Percom's MINIDOS EPROM stays on the board and provides the basic disk drivers and utility routines. In addition to the 1K on DIXIE's EPROM, another 1/2K of program sits on the disk, automatically read by DIXIE into its directory buffer area as needed. This part is called the 'transient driver'.

DIXIE requires a standard LFD-400 system with version 1.4 MINIDOS, a MIKBUG-compatible monitor, and about 3/4K of RAM just above the monitor scratchpad at A080. Any system which can run MINIDOS-PLUSX can also be used for DIXIE.

DIXIE is remarkably compatible with the basic Percom disk system and its software. Except for the fact that a DIXIE disk directory is different from a MINIDOS-PLUSX directory, each DOS can read the other's files. Thus disk conversion is not very difficult and a certain degree of compatibility exists as long as you have a printout of the disk contents. But the two operating systems cannot write on each other's disks.

(DIXIE even fits into disk systems which have Star-Kits' P-DOS Patch ROM, which adds ead-after-write error detection and correction to the Percom system. With Percom's MINIDOS in ROM socket 1 of the controller, DIXIE in socket 2, and P-DOS in socket 3, the system just purrs along with all three ROMs working together although they came from different suppliers.)

Disk Organization

Percom's MINIDOS-PLUSX uses the first two sectors of track 0 for a directory, and leaves the other eight sectors blank (file storage starts at sector 0 of track 1). DIXIE, on the other hand, makes use of all ten sectors of track 0.

Before a disk can be used under DIXIE, it must be initialized with the INITDISK utility. This utility initializes track 0 of the disk in DIXIE's special way. Thus the disk contains part of DIXIE and a variety of additional information, yet has the same storage capacity as a disk formatted by an unmodified MINIDOS-PLUSX.

Initializing a disk writes the following information on track 0: Two sectors hold the DIXIE transient driver. This 1/2K program segment loads over the directory area as needed to expand the length of DIXIE yet conserve on ROM space.

Two sectors contain a sector map and disk label. The sector map contains either 350 or 400 bytes (depending on whether a 35- or 40-track drive is used), where each byte corresponds to one sector of the

disk. If a given byte has a 0, that means the corresponding sector is empty. A non-zero number signifies that the sector holds part of a file, and the number in the byte is the number of the file (which is keyed to the file number in the directory).

Here is a major difference between disks written under MINIDOS-PLUS and DIXIE. MINIDOS-PLUSX disk files always occupy a group of consecutive sectors. Although MINIDOS maintains forward and backward pointers on each sector so that theoretically files do not have to be in consecutive sectors, in practice MINIDOS-PLUSX will always place them that way. Moreover, when a file is deleted, the released sectors are added to the free space only if the file was the last on the disk. Otherwise the released sectors are left blank.

In DIXIE, however, released sectors are marked with a 0 in the sector map, and the very next SAVE command can re-use them even if they are in the middle of the disk. Moreover, a file can be split up into non-consecutive sectors if it does not fit into a single available space. DIXIE simply searches the sector map sequentially for empty sectors, saves the file into them, and lets the standard MINIDOS backward and forward pointers tie the sectors together into a chain.

Thus DIXIE provides the same dynamic allocation scheme as Flex, SSB DOS, and other advanced systems do, but in a different way. In all of these, files can be spread out over many non-contiguous sectors on the disk. In all of these, the directory points to the first sector of each existing file, while each sector points in turn to the next sector in the chain. The last sector has a pointer of 0, which signals the end of the chain. When a file is read sequentially, the DOS follows the pointers to the end.

The difference between DIXIE and other 6800 disk operating systems is in its handling of the empty sectors. MiniFlex, Flex, and SSB DOS treat the empty sectors as just another file which happens to consist of garbage. The directory points to its first sector, and each sector points to the next empty one. When a file is deleted, its sectors simply get added at the end of the string. After many deletions and additions, the chain of empty sectors can wander back and forth over the disk. When a new file is written, the disk head may have to go in and out many times to follow the chain.

In DIXIE, on the other hand, there is no chain of empty sectors. Instead, empty sectors are flagged in the sector map with a 0. When a file is deleted, its sectors are flagged with a 0 and simply added to other adjacent empty sectors. Then, when a new file is written, DIXIE starts grabbing unused sectors, starting at the outside tracks and going in. Thus reading or writing a long file is inherently faster, since the amount of head seeks is minimized.

Similar time savings exist in other operations. For example, when Flex deletes a file, it must step the head to inner tracks of the disk in order to update pointers. DIXIE, on the other hand, just changes the sector map and rewrites it.

Since the sector map uses just part of two sectors, DIXIE also stores a disk label into the same area. Whenever a disk is initialized, the disk is given a number and name; each time a drive is accessed, DIXIE prints out this information to confirm that the correct disk was mounted.

The remaining six sectors of track 0 hold the actual directory information. There is a main directory and up to 14 lower-level catalogs. Whenever a file is saved on a disk, it is either assigned to one of the 14 catalogs, or uncatalogued. The disk contains a one-sector catalog directory which lists the names of all the catalogs, and a five-sector file directory which holds the names of all files, their file number (which is keyed to the sector map) and their catalog number (which is keyed to the catalog descriptors, and indicates which catalog holds that file). A file assigned catalog number 0 is assumed to be uncatalogued.

When DIXIE's F (Files) command is used, the directory printout provides the listing of all catalog names, as well as a listing of uncatalogued files. When the F is followed by a catalog name, then only the files belonging to that catalog are listed. Hence a disk directory listing can contain just selected classes of files.

This is a concept which is borrowed from much larger systems which may use multi - megabyte hard disks. The idea is that all Basic programs can be assigned to one catalog, all assembly language source programs to another, object codes to a third, and so on. Utilities which are commonly used would be uncatalogued and accessible outside each catalog. (The next extension would be to assign access codes so that each user could have access only to specific catalogs, but all users could access uncatalogued files.) In this way a directory printout would contain only files of interest, rather than all the (possibly hundreds) of files that a hard disk can hold.

While elegant, this idea has limited use with single-density, single-sided mini-diskettes. Since such diskettes have limited storage, are easily interchanged, and are even quite cheap, most disk users have long ago learned to devote different disks to different types of files. But if DIXIE ever gets adapted to a hard disk...

DIXIE Commands and Utilities

DIXIE has the following commands: C - Create a file. Puts the file name into the main directory or into one of the catalogs, but does not assign any disk space.
D - Create a catalog. Assigns the catalog name, but leaves it empty.
F - Files. Prints a disk directory or directory of a catalog. The directory contains the catalog or file name, file number, file size and file type, beginning, ending and transfer addresses, and first disk sector.
J - Jump to a memory address.
L - Load a file but do not run. A file is generally loaded into the same area it came from, but can be loaded into a different location.
N - Rename a file.
O - Rename a catalog.
P - Protect or un-protect a file.
A protected file can be read or renamed, but not deleted or rewritten.
Q - Query a disk. Prints out disk name and number, and number of free sectors.
R - Delete a file and release its name.
T - Delete a catalog and all its files (unless protected).
S - Save contents of memory (and a transfer address) to a disk file.
Whenever a file is saved whose name already exists on the disk, the prior file is removed (unless protected) and the new file substituted (even if it is larger or smaller; only as much space as is needed is allocated.) Note that the file must be created with the C command before it can be written to, a minor inconvenience which takes a bit of getting used to.

When any other entry is typed, DIXIE searches the disk for a file by the same name and executes it if present. Thus user programs, or disk - resident utilities are executed by typing their name.

(A complete file description consists of
#drive-number /catalog-name file-name .
where the #drive-number can be omitted for drive 1,

and the /catalog-name is omitted for uncatalogued files. The catalog-name and file-name can be up to 12 characters long and contain letters, numbers, and most punctuation symbols.)

The DIXIE EPROM comes with three disks which contain source and object code for DIXIE itself, for several disk - resident utilities, and for patches to other programs. The disk - resident utilities include INITDISK for initializing a disk, NEWLABEL for changing a disk label, SINGLECOPY and DUALCOPY for copying entire disks using one- or two-drive

systems, FILECOPY for copying single files, and PDIXD for printing a formatted disk directory on the terminal or a printer.

Program Patching

Since DIXIE files are dynamically allocated space as needed on a disk, with consecutive sectors linked via the MINIDOS forward and backward pointers, they can be read by either MINIDOS or MINIDOS-PLUSX (assuming that the starting sector of a file is known.) But they cannot be written that way since neither MINIDOS nor MINIDOS-PLUSX has any way of knowing how many sectors are free, or whether a given sector is assigned to a file or not. Hence any Percom program that writes on the disk must be patched to use named files via DIXIE.

Supplied with DIXIE are patches for Percom's HEXLDR, Assembler, and Touchup editor, as well as for the cassette versions of the TSC Editor and TSC Assembler, and for SWTP cassette Basic version 2.0 and 2.2; the latter allows program storage but not disk files. (An essential patch for Percom Super Basic is in the works and should hopefully be available soon.)

When this is done, these programs prompt for a file name (preceded by drive number and/or catalog name, if applicable), and then print out the disk label to make sure the correct disk is used. If desired, the standard DSSS Percom format can also be used for input files, so that files can be read from unformatted disks, or from MINIDOS-PLUSX - formatted disks.

Some Percom programs, such as the text processor, or DSKMAP, will work without patching since they only read files using the DSSS format. Since the DIXIE directory printout gives the starting sector for all files, this information can be given to these programs.

Those Percom programs (or programs patched by Percom) which write on a disk but use only DSSS format can still be used, but only via MINIDOS itself, not DIXIE. This might include A/BASIC or a disassembler, for instance. A user would have to be careful to use un-formatted disks (not DIXIE formatted disks), and could then transfer resulting files to a DIXIE disk with the FILECOPY utility. This process is inconvenient, but at least tolerable if it is not done too often.

Overall Conclusions

DIXIE is quite reasonably priced at $60 for the 2708 EPROM, three disks of code and patches, and a quite comprehensive user's manual. The manual contains complete instructions for using DIXIE and applying the supplied patches to existing software, as well as a good description of how DIXIE works. Since all source code is supplied on diskette, it should be easy to study or modify the system if desired.

The only question each user has to answer for himself is whether the conversion from MINIDOS-PLUSX to DIXIE will justify the time expended in translating disks and learning a new system. (Larry Preston plans to produce a utility to convert MINIDOS-PLUSX disks to DIXIE format, but it is not available at this time.)

With the exception of Percom Super Basic, other Percom programs running under MINIDOS-PLUSX use the DSSS format rather than named files. This means that a slip of the finger, or a slight mis-calculation, can store one file on top of another.

DIXIE to a large extent eliminates that possibility - at least for those programs which are patched to run under DIXIE. With DIXIE's dynamic allocation of disk space, one named file should never overwrite another. On the other hand, unpatched programs which still use the DSSS format require extra special care now, since DIXIE files can be spread out over many sectors and tracks, in places where you'd least expect them.

For many years, proponents of other disk operating have looked down at the Percom disk system and called it a fast cassette system rather than a real disk. They

should look at DIXIE, for it gives them most of the features they have always been missing, and does so at a fair price..

XREF

Not part of DIXIE, but also available from Blue Hat Software is XREF, a patch to the Percom Assembler to produce a label cross-reference printout at the end of assembly. XREF is enabled by the OPT XRF statement (which is selected by default), and disabled by OPT NOX. When enabled, the assembler assigns a line number to each line of program, and at the end prints a listing which shows for each references label (a) the line in which it is defined, and (b) all the lines in which it is used. At $15 for a disk which contains source and object code for the patch as well as a manual, XREF is a valuable addition for the heavy assembly language programmer.

CONTEST Notes

SOME NOTES ON SOFTWARE CONTEST

The following was attached to one of the contest entries. "This program hasn't been previously published in any commercial magazine or journal and is the "soul" property of xxxxxxxxxx xxxxxxxxxxxxx". Note: the ("'s) are added. Such were the feelings of one contest entrant. No doubt most all who entered had some feeling about his or her entry. After looking over a majority of the entries I must admit that I can fully understand the above.

When it all started, nearly a year past, I just did not know what to expect. A good thing, if I had known, I probably would have had second thoughts. Now however that it is 'nearly' finished, I feel a lot better about the contest and the interest it generated. Not all categories were sufficiently filled with winners (or entries) for all the prizes. A couple had an overflow of entries. Also some interesting percentage figures emerged as a result of the number of entries, per category. All in all it was enlightening and gave me a good overview of what we should do, in case we ever try another contest.

One thing became apparent, especially as things begin to drag a bit. First, I began to realize (should have known from the many buckets of sweat I have shed on some programming project) that, in many if not most cases, the 'special' beauty of a program is in the eyes of it's creator. As the local judges (we used 6 local 68XX users to judge portions of the entries) came together, after reviewing their respective

charges, each with some decisions made, it became apparent that not everyone views the merits of a particular program to the same degree. In fact it was difficult, at first, to believe that we would ever get it finished. We even nit-piked some. I stayed out of all final decisions. If I felt that some particular program was lightly or heavily dealt with, I would gather another committee, of the judges present, or occasionally had to get on the phone to some that were not present), and have them give a second opinion. Sort of like a medical conference. We would take their opinions back to the rest of the group, again sometimes by phone, and finally each entry fell into it's place of judgement. I can tell you this, none of us agreed that the final results were as we would have personally preferred, however, I honestly believe that the decisions were as fair as any committee could arrive at. Such is the penalty of group action; honest, fair but cumbersome.

Anyway we have all the local judging finished and are waiting for two categories, being judged in different parts of the country, to be returned. This means that I can get it finally wrapped up (?). The thing that I hope does not occur is that any one gets his or her feelings hurt. I am sure that some who ended up in one position could have well been in another (better or worse) position, had other judges evaluated their offering. I have even received phone calls and letters from friends of some who entered, extolling the virtures of a particular entry, their friend's. For this reason, as well as others, I had no hand in any FINAL decision. When the final results are published, hopefully next month, the die will be cast. No appeals or reviews, right or wrong, the judges decisions will stand!

More will be expounded upon next month (?), but for now I just want to heartfully thank all who entered and all who supported this effort by their donations. I honestly believe that we are all a little closer as a result of those involved.

As for the prizes, excluding those from 68 Micro Journal (subscription winners to be notified from here), letters will be mailed to all donors requesting that they forward their particular prize on to the winner. Also the occasional use in this column of the character (?) is mainly due to the feeling I had, at the beginning, that we could get it finished in 4 or 5 months. Oh well, what was it some one said about 'the best laid plans of mice and (?)'.

FLEX User Notes

BY: RONALD W. ANDERSON
3540 STRUBRIDGE COURT
ANN ARBOR, MI 48105

One of the most prolific of the suppliers for 6800/6809 software is Technical Systems Consultants of West Lafayette Indiana. One of their newer offerings is their Flex Diagnostics Package for 6809. This package includes a series of memory test programs and a group of programs for checking, diagnosing and repairing disks that have "crashed". The manual that comes with the programs is 100 pages long. Though TSC has always produced a better than average manual, their trend is to better and better documentation. This time they have done extremely well. The manual is much more than instructions for the use of the diagnostic programs. It contains long discussions of memory organization and of the types of failures that may be encountered. It points out that dynamic memory has different modes of failure than static memory, which is the reason for the inclusion in the package of a dynamic memory test.

The diagnostics are roughly divided into two equal groups. The first is the memory test group. It is pointed out in the manual that different modes of failure are detectable by different test methods, and the manual recommends the order in which to try the test programs to try to find a bad memory chip. The tests are listed here with a description extracted from the TSC information.

CONVERGE - used to detect address or data lines that are shorted together.

DYNAMIC - for testing of dynamic memory of course.

QUICK - a check that reveals "solid" failures. It runs very fast.

RANDOM - a slower test that shows up more subtle failures. Essentially test of memory with changing random pattern.

WALKO - test in which only one bit in the memory being tested is a 0.

WALK1 - inverse test of WALKO.

Each of these tests takes longer than the previous one in the list. Some of you may have the TSC diagnostics package that is compatible with the AC-30 cassette system. These tests are similar in result, but they are completely new in terms of operating with the disk system, messages that they present, and in the ease of escape from the tests back to FLEX. All of the tests recognize when FLEX has been overwritten, and warn you of the fact, then exit to the monitor when they are finished.

The other half of the diagnostics as I said above, is the Disk Diagnostic Package. Here again, TSC has provided a short textbook on how a FLEX disk is organized, and what some of the failure modes are. There are 10 programs for diagnosing and solving disk problems in the package. TEST is a fast program that checks for bad spots on the disk. FILETEST does the same thing but indicates in which files the bad spots are located. VALIDATE looks for discrepancies between the file and the information in the directory, intersecting files, etc.

Four of the programs are used to recover data from damaged disks. TSC does stress the fact that a good backup is the best way to recover from a disk failure, but the utilities given here allow for the recovery of a single file, or a try to reconstruct all the files on a disk, even if the directory is "wiped out". REBUILD and RECOVER are for the recovery of files from such a disk. REBUILD will attempt to save all the files from such a disk, while RECOVER allows specification of which file to save, however, requiring that you specify the starting track and sector of the file. RAWCOPY will copy a file with a checksum error, to a new file. This would be of most use in recovering a text file. The damaged area will still be damaged, but the text editor may be used to repair this area if it is not extensive. UNDELETE does just what it says. When a file is deleted, the sectors that it occupied go back into the free chain at the end. Therefore, the most recently deleted file becomes the last area to be written over. There is therefore a reasonable chance that the last few files deleted are still intact, particularly if the disk is not too full. This utility will indicate all the intact but deleted files and allow you to "dump" them to the terminal in an effort to determine which is the one of interest. You may then recover any of them, specifying a filename for the "undeleted" file. I know of one case where a few hours of typing could have been saved with such a command. An operator was using the text editor and processor, and had been working for some time on a file when the power failed. When power was restored, the o erator tried to edit the backup file but mistakenly didn't rename the file from .BAK to TXT. When the editor asked "DELETE THE BACKUP FILE", the operator responded with a Y, and deleted the backup. The new file was not written when the power failed and the file was lost. Not only did the last edit session have to be redone, but a whole large file had to be retyped. UNDELETE would have allowed recovery of the backup at least, and only the last edit would have been lost.

COPYR is a s ecial utility used to rebuild a sector map for a random file that has been recovered. FLAW is used to remove an

"intermittent" sector from the sector map of the disk so that the remainder of the disk may be used. EXAMINE is in a class by itself. The Flex Newsletter 2 from TSC had a "Dump and Repair" utility. My newsletter had a FILEPAT utility. This one is a combination of the two and about 4 more. It allows you to look at any sector on the disk, and make byte by byte changes. It does this by reading the sector to the sector buffer in the FCB, and allowing you to edit the information there with a function like the memory examine and change function in SWTBUG and SBUG-E. No changes are made to the disk until you have had a chance to make the desired changes and dump the sector to the screen again to see if all is correct. You may then write the sector back to the disk or abort the change. There are many ways of moving from sector to sector on the disk. The user is fully prompted for the required input, and like all the other utilities of this package, there are numerous messages generated by this utility.

Those of you who have read my reviews before in other publications and my newsletter, know that I am not always totally c plementary of software that I review. I would have to try very hard to find something to complain about. Perhaps it could be said that the price for the package seems kind of steep for the number of sectors of program on the disk. On the other hand, someone said that good things come in small packages. If you can recover a disk file, or find a memory problem and get your system back running with this set of utilities, perhaps you will save the cost of them the first time you have to use the package! We all know that we should keep backups of our disks, but how many of us back up our system disk every time we add a utility or make an improvement in, say our startup file? I find out how sloppy I have been when I lose a file and the backup has 40 sectors less than the disk it is backing up! I can't give this one less than an AAA rating, both for the documentation and the programs.

MORE ON PRINTER MODES

In an earlier issue, I published the "U" utility group. In the discussion of its uses, I mentioned the H sub-utility being used to output control characters to the printer. This month, you will note the listing of the SET utility. While this is written expressly for the Paper Tiger, I am sure that any of you with a little Assembler programming experience could use it as a model for another set of control characters. As in the U utility last time, I've used a set of single letter commands that follow the filename SET on the command line, to provide for setting the various modes of the printer. I've tried to use letters that could represent the various modes, such as X for eXtra wide, s for Skinny, etc. There is also a C for Clear, that returns the printer to its normal status. This utility assumes that you have previously loaded PRINT.SYS either by having used a command with a P, preceeding it, or by using GET PRINT.SYS. It checks to see that PRINT.SYS is loaded and issues an error message if not. This means that you don't have to use P before SET. SET,W will set the printer to 10 characters per inch. Note that these work in a TSC BASIC program also, using the EXEC command. This means that you can change your printer mode under program control in a BASIC program.

Incidentally, there is one sure way to have previously loaded PRINT.SYS when you use the SET utility. Your STARTUP.TXT file could contain the line GET PRINT.SYS.O. Whatever combination of features your printer has, you may make them avilable to you at the DOS level, and to all your TSC BASIC programs. You will notice that the same technique was used here as in last month's U utility for finding the proper entry point in the program from the command letter. The letter is compared to the first of each three byte group in the JTAB at the end of the program. If and when a match is found, the starting address for that segment of the program is found in the next two bytes, by a LDX 1,X instruction, and then executed with a JMP 0,X. Several bytes of program were saved by using a common output character routine at the label OUT. All of the set commands branch to this point after loading the A Accumulator with the proper control character. Here, there is a JSR POUT, to the print output routine, and a JMP WARMS to get back to FLEX.

You will note that I have used a FLEX2 equate file for the equates. Simply by using a FLEX9 equate file, this program could be assembled with the 6809 Assembler, and would run in the '09 system as well. As I am writing this, it occurs to me that this is the only necessary change to get FLEX2 utilities to run in 6809. Of course, the '09 utilities ORG at $C100 and the FLEX2 utilities at $A100 so that the ORG would have to be changed. TSC supplied manual on the Assembler describes the Library feature, as did this column last month. The library file does not, in fact must not have an END at its end, or the Assembler will stop assembling at the end of the library file. The library file must contain just the lines that you would normally want inserted at the point where the LIB instruction is inserted in the program.

I thought it might be informative and educational to present the SET program here in Pascal and BASIC. First of all, it does not make sense to write the program in a BASIC interpreter, because then you'd have to load BASIC just to set the printer mode. Some of you have Microware's A/BASIC compiler, and for those of you who don't, it is a subset of BASIC, and the exercise might be informative. The largest limitation to A/BASIC is (at least for present purposes) that it does not allow a statement after an IF - THEN. You may only go to another line or to a subroutine. I have written two programs to show some alternative ways of using BASIC. The problem to be solved with this program is one that may be generalized to a type called an N way branch. We want to examine an input to see if it is one of several possible values, and take a different ection for each possibility, including reporting an error if it is not one of the valid inputs.

The program SETBAS uses a peek and poke to control the output such that the prompt for the command goes to the terminal, but the output of the control character is to the printer. You must use the P, preceeding this program, ie. P,SETBAS. The program prompts for the command letter, outputs it to the printer and returns to FLEX. The letter input in response to the prompt is simply tested in turn for each of the valid values, and a jump is made to the proper output response line when a match is found. If no match is found, an error message is printed and the prompt is output again.

SETBAS1 is an alternative way of doing this comparison. It uses the "if not then skip" approach. The result is a little cleaner, but not any easier to follow. What is needed here is a way of making a multiple choice. There is the ON GOTO statement in BASIC, but that only works on consecutive numbers starting at 1. WE could use a series of statements in A/BASIC, that would assign a value to a variable Q, such as (rather then line 51 being PRINT CHR$(28)), let line 51 be Q=1, etc. for the other values of M$. Then, you could ON Q GOSUB 60, 61, 62, 63, 64, 65, 66, 67. These lines could print the various control characters. This arrangement, while a little more structured, would mostly serve to make the program a little longer. Of course, in a BASIC that allows statements after an IF - THEN, you could simply use 7 lines such as IF M$="X" THEN PRINT CHR$(28). This would be a bit simpler because the number of GOTO's would be greatly reduced.

Pascal has a simpler solution, or at least one that is more structured. The Pascal CASE construct is just like the ON GOTO, in that it is our N way branch. Pascal, however, allows integers, Characters, or T PE variables in the CASE statement. Without getting into the subject of enumerated, scalar, or TYPE variables in Pascal, the Character variable use here will work fine. It is also possible in Pascal to define constants with names just like variable names in BASIC. The names or identifiers as they are called in Pascal, may be long, but only the first 6 characters are used by the program. (This is a limitation of Lucidata's implementation of Pascal, and not necessarily true of all implementations). By assigning values to names, the program makes more sense when we try to read it. This Pascal also allows hexadecimal constants. They are identified as in the Assembler, with a dollar sign. It makes more sense to write POKE (SWITCH, ON), than to write POKE (-21470, -1), though they are equivalent.

You will note that the control characters have been defined too. WRITE is equivalent to PRINT in BASIC, and READ is equivalent to INPUT. The CASE statement is almost self explanatory. Which ever "label" matches the input character, its corresponding statement immediately to its right, is executed. All CONSTants and VARiables must be declared at the start of the program, and their types given. An attempt to use a varible of the wrong type, eg. to add a character to a number, will result in an error when the program is compiled. All the Variable declarations may look like a pain when you start using Pascal, but you will soon find that they allow the compiler to catch many of the "dumbs" that we all build into a program. By the time your program has made it through the compiler with no errors, it will be pretty close to running properly, though not necessarily error free.

THE 68000 PROCESSOR

Please consider this to be my personal opinion. I will be glad to receive contrary opinions from readers. Today, I attended a Motorola Seminar on the 6809 and 68000. The emphasis was on the 68000. Motorola has a great deal at stake in this development. They now have an EXORmacs development system for the 68000, though there are still available a number of cross assemblers and compilers to run on other computers, including the Exorciser 6800 system. The 68000 is a 16 bit processor with 32 bit capabilities just as the 6809 is an 8 bit processor with 16 bit capabilities. Motorola has extensive plans for a line of "Micromodule" cards for the 68000, though they are 14 1/2 inches by 9 1/2, so they don't look very "micro". Incredibly (to me) they are still not providing any high level software with floating point arithmetic capabilities. It is certainly because of the applications in which I am involved, but I can't see why anyone would need a processor as powerful as the 68000 if they were not doing calculations of some magnitude. They are planning a version of Pascal with REAL (floating point) variables if they see a demand. (Interestingly only about 5% of those attending indicated a need for it). Motorola has a PL/M for the 6800 and 6809, and will extend it to the 68000 also, but it has only integer arithmetic too. They do have a very powerful assembler in the mill for the 68000 with (strangely) a few of the higher level constructs such as DO WHILE, REPEAT UNTIL, IF THEN, etc. These will take care of loop construction in the assembler, and make the loop exit condition tests work correctly for the programmer. They admit that this is somewhat a reversal of the normal philosophy, but it sounds like it may be a very good feature.

All the software that is planned for the 68000 is planned in a 6809 version too, so there will be a great deal of support for the '09. In addition to the software support, there will be a lot of new peripheral chips for use in larger systems. There is a memory management chip, a combination PIA - timer, DMA controllers, etc. The 68000 has many features that were put there to make it easy to implement higher level languages. There is in addition to a 16 bit multiply, a divide instruction. In fact, there are signed and unsigned multiply and divide instructions. Instructions work at the bit, byte, word (16 bit) and double word (32 bit) level. A friend who works with an LSI-11 indicates that the two are very similar in capabilites and instruction sets. IT will probably be one to two years before any of this filters down to the hobbyist level. Some of the chips and software are a year away as of this writing (May). The 68000 sold for $250 a few months ago, but the current price is $180. What "scares" me is all the hardware support required. The 68000 can address 16 Megabytes directly with its 24 bit address bus. The data bus is 16 bits, but N by 8 memory can be accomodated, and is planned for at this point.

I guess, to sum up my impressions, very few of the people at the seminar, and I have no idea how many are currently 6800/6809 users, are using the current devices to a small fraction of their capabilities. Perhaps there were many present who don't now use Micros. It would seem to me that more would be accomplished by using the 8 bit micros capabilities, and pushing these a little before jumping into twice the hardware, four times the complexity, and who knows how many times the cost, to do the same job. With 16 Megabytes of addressable memory, who will bother to write efficient compilers? The end of all this will be tremendously complex systems that no end user will be able to understand or maintain. The large increase in hardware will result in a large decrease in reliability at the same time. Don't get me wrong, I think there are good applications for 16 bit machines, and even the 32 bit processor that Motorola is undoubtedly working on, but these are not the same applications for which the 8 bit machines are ideal. Motorola should be aiming the larger processors at the market in which they are necessary to do the job. For example, complex NC machine controls, not those in which each move is programmed by a punched tape, but those into which the part parameters are programmed and the motions are calculated in real time. Here speed is important. Some of the present machines are rate limited by the calculation time. How about Business applications? If Radio Shack can sell a TRS-80 system for a small business, and they do seem to be doing so with some good results, how about a 68000 system for a medium sized business? How about a time sharing system for, say, a college Chemistry Department? One of these systems would be ideal for a medium sized Engineering Department for design work. All these applications imply the availability of fully implmented or nearly fully implemented versions of the common programming languages.

Now that I've climbed on my soapbox for this time, I'll get off, and see what kind of response I get. See you all next time.

A final note, apparently Lucidata Pascal does not use the normal FLEX routines PUTCHR and GETCHR for its I/O, and the SET program in Pascal, though it does set the printer mode properly, does not switch the prompts to the terminal as planned. Pascal has an overlay to tie in a printer, which I have not yet implemented. The program serves as a better example as is, since it impliments the same functions as the one in A/BASIC and the assembler version.

```
MODEM INPUT AS A DISK TEXT FILE      6-16-80  TSC ASSEMBLER  PAGE    1

                         *
                         *
                         * TERMEM OPENS A DISK FILE SPECIFIED IN THE COMMAND LINE AND
                         * INPUTS ALL INPUT FROM THE MODEM (INCLUDING THE ECHO OF YOUR
                         * COMMANDS INPUT FROM YOUR TERMINAL) TO MEMORY.  WHEN YOU TYPE
                         * AN ESCAPE CHARACTER, THE DATA IS SAVED TO THE DISK FILE.
                         *
                         * TERMEM COMES UP IN THE NO-ECHO MODE.  IF YOU WANT TO SWITCH
                         * THE ECHO MODE TYPE CONTROL E TO TOGGLE ECHO ON AND OFF.
                         *
                         * EXAMPLE:   TERMEM.TEST
                         
                         * THIS WILL OPEN A FILE TEST.TXT ON THE WORKING DRIVE AND SAVE
                         * THE COMMUNICATION TO IT.
                         *

                         * EQUATES

     0000                BEGMEM  EQU    $000
     7FFF                ENDMEM  EQU    $7FFF      ALTER THIS TO REFLECT YOUR SYSTEM MEMORY END

     8000                PORT0   EQU    $8000
     8004                PORT1   EQU    $8004

     A840                FCB     EQU    $A840      FLEX2 EQUATES
     A00C                PORECH  EQU    $[illegible]  SWTBUG EQUATE
     AD03                WARMS   EQU    $AD03
     AD15                GETCHR  EQU    $AD15
     AD1E                PSTRNG  EQU    $AD1E
     AD24                PCRLF   EQU    $AD24
     AD2D                GETFIL  EQU    $AD2D
     AD33                SETEXT  EQU    $AD33
     AD3F                RPTERR  EQU    $AD3F

     B403                FMSCLS  EQU    $B403
     B406                FMS     EQU    $B406

     A100                        ORG    $A100      FLEX UTILITY SPACE

     A100 20 05          TERM    BRA    TERM2
     A102 01             VN      FCB    1
     A103                BEGADR  RMB    2
     A105                ENDADR  RMB    2


     A107 CE A8 40       TERM2   LDX    #FCB
     A10A BD AD 2D               JSR    GETFIL     GET FILE SPEC FROM COMMAND LINE
     A10D 24 07                  BCC    TERM3      IF NO ERROR
     A10F 86 15                  LDA A  #21
     A111 A7 01                  STA A  1,X
     A113 7E A1 FF               JMP    ERROR

     A116 CE A8 40       TERM3   LDX    #FCB
```

```
A119 B6 01                 LDA A  01
A11B BD A0 33              JSR    SETEXT    TO .TXT

A11E CE A8 40    OPEN      LDX    #FCB
A121 86 02                 LDA A  #2        OPEN FOR WRITE
A123 A7 00                 STA A  0,X
A125 BD B4 06              JSR    FMS
A128 27 03                 BEQ    ACINIT
A12A 7E A1 F9              JMP    FILERR

A12D CE 80 08    ACINIT    LDX    #PORT0
A130 86 03                 LDA A  #3
A132 C6 15                 LDA B  #$15
A134 A7 00                 STA A  0,X       RESET ACIA
A136 E7 00                 STA B  0,X       SET UP CONTROL BITS
A138 A6 01                 LDA A  1,X       CLEAR RECEIVE BUFFER

A13A CE A2 8D              LDX    #MSG4     READY MESSAGE
A13D BD A0 1E              JSR    PSTRNG
A140 BD A0 24              JSR    PCRLF
A143 CE 00 00              LDX    #BEGMEM
A146 FF A1 03              STX    BEGADR
A149 FF A1 05              STX    ENDADR
A14C 7F A0 0C              CLR    PORECH
A14F 73 A0 0C    TERMEC    COM    PORECH    INITIALIZE OFF

                 * MAIN INPUT LOOP HERE

A152 CE 80 08    TSTMDM    LDX    #PORT0
A155 E6 00                 LDA B  0,X
A157 54                    LSR B
A158 24 1D                 BCC    TSTTRM    TEST FOR RECEIVE BUFFER FULL

A15A A6 01                 LDA A  1,X       HAVE CHARACTER FROM MODEM
A15C 84 7F                 AND A  #$7F      GET RID OF PARITY BIT
A15E CE 80 04              LDX    #PORT1
A161 BD A1 9E              JSR    OUTCHR    OUTPUT TO TERMINAL
A164 BD A1 A7              JSR    MEMORY    PUT IN MEMORY
A167 24 03                 BCC    MDM1
A169 7E A1 BF              JMP    MEMOV
A16C 7D A0 0C    MDM1      TST    PORECH    ECHO ON?
A16F 26 06                 BNE    TSTTRM
A171 CE 80 08              LDX    #PORT0    SET UP ECHO
A174 BD A1 9E              JSR    OUTCHR    GO DO IT

A177 CE 80 04    TSTTRM    LDX    #PORT1    SEE IF RECEIVE BUFFER FULL
A17A E6 00                 LDA B  0,X
A17C 54                    LSR B
A17D 24 D3                 BCC    TSTMDM    GO AROUND AGAIN IF NO CHARACTER

A17F A6 01                 LDA A  1,X
A181 84 7F                 AND A  #$7F      GET RID OF PARITY
A183 81 1B                 CMP A  #$1B      IS IT ESCAPE ?
A185 27 3E                 BEQ    EXIT

A187 81 05                 CMP A  #5        IS IT ^E ?
A189 27 C4                 BEQ    TERMEC    GO CHANGE ECHO MODE

A18B CE 80 08              LDX    #PORT0
A18E BD A1 9E              JSR    OUTCHR    OUTPUT TO MODEM
A191 7D A0 0C              TST    PORECH
A194 26 BC                 BNE    TSTMDM

A196 CE 80 04              LDX    #PORT1
A199 BD A1 9E              JSR    OUTCHR    ECHO
A19C 20 B4                 BRA    TSTMDM

                 * SUBROUTINES

A19E E6 00       OUTCHR    LDA B  0,X
A1A0 54                    LSR B
A1A1 54                    LSR B
A1A2 24 FA                 BCC    OUTCHR    WAIT FOR TRANSMIT BUFFER EMPTY
A1A4 A7 01                 STA A  1,X       OUTPUT IT
A1A6 39                    RTS

A1A7 FE A1 05    MEMORY    LDX    ENDADR
A1AA 81 08                 CMP A  #$08      BACKSPACE?
A1AC 27 04                 BEQ    MEM1

A1AE A7 00                 STA A  0,X
A1B0 08                    INX
A1B1 08                    INX
A1B2 09          MEM1      DEX
A1B3 FF A1 05              STX    ENDADR
A1B6 8C 7F FF              CPX    ENDMEM
A1B9 27 02                 BEQ    MEM2
A1BB 0C                    CLC
A1BC 39                    RTS
A1BD 0D          MEM2      SEC
A1BE 39                    RTS              SET MEM OVERFLOW FLAG

A1BF CE A2 37    MEMOV     LDX    #MSG1
A1C2 BD A0 1E              JSR    PSTRNG
A1C5 FE A1 03    EXIT      LDX    BEGADR
A1C8 BC A1 05              CPX    ENDADR
A1CB 27 20                 BEQ    CLOSE     DON'T SAVE EMPTY FILE!

A1CD A6 00       SAVE      LDA A  0,X
A1CF 08                    INX
A1D0 84 7F                 AND A  #$7F
A1D2 81 20                 CMP A  #$20
A1D4 24 04                 BCC    SAVE1     CONTROL CHARACTER
A1D6 81 0D                 CMP A  #$0D
A1D8 26 0E                 BNE    SAVE2
A1DA FF A1 03    SAVE1     STX    BEGADR
A1DD CE A8 40              LDX    #FCB
A1E0 BD B4 06              JSR    FMS
A1E3 26 1A                 BNE    ERROR
A1E5 FE A1 03              LDX    BEGADR
A1E8 BC A1 05    SAVE2     CPX    ENDADR
A1EB 26 E0                 BNE    SAVE

A1ED CE A8 40    CLOSE     LDX    #FCB
A1F0 86 04                 LDA A  #4        CLOSE FILE CODE
A1F2 A7 00                 STA A  0,X
A1F4 BD B4 06              JSR    FMS
A1F7 27 0C                 BEQ    EXITE
A1F9 A6 01       FILERR    LDA A  1,X
A1FB 81 03                 CMP A  #3
A1FD 27 0C                 BEQ    ASKDEL

A1FF BD AD 3F    ERROR     JSR    RPTERR
A202 BD B4 03              JSR    FMSCLS
A205 7F A0 0C    EXITE     CLR    PORECH
A208 7E AD 03              JMP    WARMS

A20B CE A2 59    ASKDEL    LDX    #MSG2
A20E 8D 1C                 BSR    ASK
A210 26 F3                 BNE    EXITE

A212 CE A2 7C              LDX    #MSG3
A215 8D 15                 BSR    ASK
A217 26 EC                 BNE    EXITE

A219 CE A8 40              LDX    #FCB
A21C 86 0C                 LDA A  #12
A21E A7 00                 STA A  0,X       DELETE FILE
A220 BD B4 06              JSR    FMS
A223 26 DA                 BNE    ERROR
A225 A6 24                 LDA A  36,X
A227 A7 04                 STA A  4,X
A229 7E A1 1E              JMP    OPEN      OPEN NEW FILE

A22C BD AD 1E    ASK       JSR    PSTRNG
A22F BD AD 15              JSR    GETCHR
A232 84 5F                 AND A  #$5F
A234 81 59                 CMP A  #'Y
A236 39                    RTS
A237 4D          MSG1      FCC    'MEMORY OVERFLOW SOME DATA LOST ! '
A258 04                    FCB    4
A259 4D          MSG2      FCC    'MAY THE EXISTING FILE BE DELETED?
A27B 04                    FCB    4
A27C 41          MSG3      FCC    'ARE YOU SURE? '
A28A 04                    FCB    4
A28B 52          MSG4      FCC    'READY FOR MODEM INPUT '
A2A1 04                    FCB    4

                           END    TERM

NO ERROR(S) DETECTED
```

```
    PROGRAM SCIPKG;

    (* SCIENTIFIC FUNCTIONS ************************************************)
        (* IF YOU USE THIS PACKAGE IN A PROGRAM, THE FOLLOWING CONSTANT
           AND  VARIABLE DECLARATIONS MUST BE MADE AT THE START OF YOUR
           PROGRAM SO THEY ARE GLOBAL FOR  ALL  OF  THE  FUNCTIONS  AND
           PROCEDURES HERE.                                                   *)

    CONST
      PI=3.14159265;
      LOG10= 2.302585091

    VAR
      X,Y,N,THETA : REAL;

    (* POLYNOMIAL APPROXIMATION CALCULATIONS FOR SINE ************)

    FUNCTION POLY(X,S0,S1,S2,S3,S4,S5,S6,S7,S8,S9,S10 : REAL) :REAL;

  BEGIN
  POLY:=((((((((((S10*X+S9)*X+S8)*X+S7)*X+S6)*X+S5)*X+S4)*X+S3)*X+S2)*X+S1)*X+S0)
  END;

    (* SINE FUNCTION ********************************)

    FUNCTION SIN (X:REAL) : REAL;

  CONST
    S0=-6.88677461E-9;
    S1=1.00000171;
    S2=-2.27738650E-5;
    S3=-0.166556219;
    S4=-2.59650290E-4;
    S5=8.65844759E-3;
    S6=-2.14944062E-4;
    S7=-1.35569080E-4;

  VAR
    NEGATIVE : BOOLEAN;
    RESULT : REAL;

    BEGIN (* MAKE X POSITIVE IF NEGATIVE *)
      WHILE X < 0 DO
      BEGIN
        X:= X + 2 * PI;
      END;

      WHILE X > 2*PI DO (* REDUCE POSITIVE X TO 2 PI OR LESS *)
      BEGIN
        X:= X - 2 * PI;
      END;

      IF X > PI THEN (* NOW REDUCE TO PI OR LESS AND GET RESULT *)
      BEGIN
        X := X - PI;
        NEGATIVE := TRUE;
      END
      ELSE NEGATIVE := FALSE;

      IF X>PI/2 THEN X:= PI-X; (* REDUCE FURTHER TO 0 TO PI/2 RANGE *)

      RESULT:=POLY(X,S0,S1,S2,S3,S4,S5,S6,S7,0,0,0,0,0);
      IF NEGATIVE THEN SIN:=-RESULT ELSE SIN:= RESULT;
    END;

    (* FOR COSINE, ADJUST X BY PI/2 AND USE SINE ***********)

    FUNCTION COS (X:REAL) : REAL;

    BEGIN
      X:= X+PI/2;
      COS:= SIN(X);
    END;

    (* ARCTANGENT USING POLYNOMIAL EXPANSION ***************)
    (* RETURNS FIRST QUADRANT ANGLE.  X MUST BE POSITIVE *)

    FUNCTION ATN(X : REAL) : REAL;
```

```
CONST
   A0=-7.84377594E-9;
   A1=1.00000081;
   A2=-2.71707901E-4;
   A3=-0.330122299;
   A4=-0.0196470651;
   A5=0.230260300;
   A6=-0.0932646250;
   A7=-0.101873740;
   A8=0.0969531201;
   A9=-0.0273848117;

VAR
   RECIPROCAL : BOOLEAN;
   RESULT : REAL;

BEGIN
   IF X >1 THEN
   BEGIN
      X:= 1/X;
      RECIPROCAL:=TRUE;
   END
   ELSE RECIPROCAL:= FALSE;

   IF X <> 0 THEN
   RESULT:=POLY(X,A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,0,0)
   ELSE RESULT := 0;

   IF RECIPROCAL THEN
      ATN:= PI/2-RESULT
      ELSE ATN:=RESULT;
END;

(* EXPONENTIAL FUNCTION E^X ****************************************)

FUNCTION EXP(X : REAL) : REAL;

CONST
  E0 = 0.998118441;
  E1 = 1.01049357;
  E2 = 0.474745507;
  E3 = 0.200839847;
  E4 = 0.0133540345;
  E5 = 0.0229112090;
  E6 = -3.07135464E-3;
  E7 = 8.90571617E-4;
  EPSILON = 2.71828182;

VAR
  N : REAL;

BEGIN
  N:=1;

  WHILE X > 2.0 DO
  BEGIN
    X:=X-1;
    N:=N * EPSILON;
  END;

  WHILE X<1.0 DO
  BEGIN
    X:=X+1;
    N:=N / EPSILON;
  END;

  EXP:= POLY(X,E0,E1,E2,E3,E4,E5,E6,E7,0,0,0,0,0) * N;
END;

  (* LOG TO THE BASE E *)

  FUNCTION LOG(X : REAL) : REAL;

CONST
  L0 = -2.94335341;
  L1 = 9.85608478;
  L2 = -20.3764221;
  L3 = 30.0265796;
  L4 = -27.0934290;
  L5 = 10.2186084;
  L6 = 6.38598514;
  L7 = -10.7526069;
  L8 = 6.27018730;
  L9 = -1.80549378;
  L10 = 0.213658986;

  VAR
    RESULT : REAL;
    N : REAL;

  BEGIN  (* ADJUST NUMBER TO BETWEEN .15 AND 1.5 *)
    N:=0;
    WHILE X>1.5 DO (* ADJUST LARGE NUMBERS *)
      BEGIN
        X:= X/10;
        N:= N+1
      END;

    WHILE X<0.15 DO (* ADJUST SMALL NUMBERS *)
      BEGIN
        X:= X * 10;
        N:= N-1
      END;

    IF X <0.475 THEN
    BEGIN
      X:=X*3.16227766; (* SQUARE ROOT OF 10 *)
      N:=N-0.5;
    END;

    IF X<>0 THEN
    BEGIN
      EXP:=POLY(X,L0,L1,L2,L3,L4,L5,L6,L7,L8,L9,L10)+N* LOG10;
    END;

  END;

  (* RECTANGULAR TO POLAR CONVERSION ****************************)
  (* MATH CONVENTION. +X AXIS 0 DEGREES. INCREASING CCW *)
  (* NOTE CORRECTIONS IN THIS PROCEDURE FOR ZERO VALUES FOR X AND Y *)

  PROCEDURE POLAR (X,Y : REAL);

  BEGIN
    R:=SQRT(SQR(X)+SQR(Y));

    IF X<>0 THEN THETA := ATN(ABS(Y/X))
    ELSE
      BEGIN
        IF Y>0 THEN THETA := PI/2
        ELSE THETA := 3*PI/2;
      END;

      IF(X<0) AND (Y>=0) THEN
        THETA:= PI - THETA;  (* SECOND QUADRANT ADJUST *)

      IF(X<0) AND (Y<0) THEN
        THETA:= PI + THETA;  (* THIRD QUADRANT *)

      IF(X>0) AND (Y<0) THEN
        THETA:= 2*PI - THETA;  (* FOURTH QUADRANT *)
    END;

  (* NOTE THIS LISTING INCLUDES ONLY PROCEDURES.  THERE IS NO MAIN
     PROGRAM AND IT WILL NOT COMPILE AS IS *)
```

```
                         NAM     P.CMD
                         TTL     P WITHOUT PRINT.SYS
                         STTL    LOAD PRINT AND INITIALIZE
                         OPT     PAG
                         PAG

P WITHOUT PRINT.SYS               9-3-80 TSC ASSEMBLER   PAGE    1
LOAD PRINT AND INITIALIZE

                    *
                    *
                    * THIS IS A SINGLE LOAD, SINGLE SECTOR PRINT COMMAND AND PRINT
                    * ROUTINE ALL IN ONE.  ADAPTED FROM A MINIFLEX VERSION FROM
                    * FRANCIS VAN HORN BY RON ANDERSON.
                    *
                    *
                    * EQUATES

               E01C PIA     EQU     $E01C
               CC02 EOL     EQU     $CC02
               CC03 DEPTH   EQU     $CC03
               CD15 GETCHR  EQU     $CD15
               CC11 LSTTRM  EQU     $CC11
               CD1E PSTRNG  EQU     $CD1E
               CD06 RENTER  EQU     $CD06
               CD0F OUTCH   EQU     $CD0F
               CD10 SWAP    EQU     OUTCH+1
               CD03 WARMS   EQU     $CD03

 C100                       ORG     $C100
 C100 20    01      P       BRA     START
 C102 02            VER     FCB     2
 C103 B6    CC11    START   LDAA    LSTTRM      GET LAST TERMINATOR
 C106 81    0D              CMPA    #$D         CR?
 C108 27    14              BEQ     P2          IF YES, REPORT ERROR
 C10A B1    CC02            CMPA    EOL         EOL CHARACTER?
 C10D 27    0F              BEQ     P2          IF YES, REPORT ERROR

                    * CLEAR DEPTH RATHER THAN PAUSE SO ESCAPE WILL HALT OPERATION

 C10F 7F    CC03            CLR     DEPTH
 C112 CE    CCE4            LDX     #POUT
 C115 FF    CD10            STX     SWAP        PUT IN OUTCH+1
 C118 BD    CCC0            JSR     PINIT       DO INITIALIZE
 C11B 7E    CD06            JMP     RENTER      RETURN CONTROL TO DOS
 C11E CE    C127    P2      LDX     #ERSTR      POINT TO ERROR STRING
 C121 BD    CD1E            JSR     PSTRNG      PRINT IT
 C124 7E    CD03    P3      JMP     WARMS       RETURN TO DOS

                    * STRINGS FOLLOW

 C127 43 4F 4D 4D   ERSTR   FCC     /COMMAND MUST FOLLOW "P"/
 C13E 04                    FCB     4

                    * STANDARD PARALLEL OUTPUT ROUTINE OR ONE FOR YOUR PRINTER
                    * THIS ONE DRIVES A PAPER TIGER WELL
                    *
 CCC0                       ORG     $CCC0
 CCC0 86    3A      PINIT   LDAA    #$3A
 CCC2 B7    E01D            STAA    PIA+1
 CCC5 86    FF              LDAA    #$FF

P WITHOUT PRINT.SYS               9-3-80 TSC ASSEMBLER   PAGE    2
LOAD PRINT AND INITIALIZE

 CCC7 B7    E01C            STAA    PIA
 CCCA 86    3E      POUTA   LDAA    #$3E
 CCCC B7    E01D    POUTB   STAA    PIA+1
 CCCF 39                    RTS
                    *
                    * CHECK FOR PRINTER READY
                    *
 CCD0                       ORG     $CCD0
 CCD0 7D    E01D    PCHK    TST     PIA+1
 CCD3 39                    RTS
                    *
                    * PRINTER OUTPUT CHARACTER ROUTINE
                    *
 CCE4                       ORG     $CCE4
 CCE4 8D    F2      POUT    BSR     PCHK
 CCE6 2A    FC              BPL     POUT
 CCE8 7D    E01C            TST     PIA         CLEAR READY FLAG
 CCEB B7    E01C    POUT1   STAA    PIA
 CCEE 86    36              LDAA    #$36
 CCF0 8D    DA              BSR     POUTB
 CCF2 20    D6              BRA     POUTA
                            END     P

0 ERROR(S) DETECTED
```

```
SET PRINTER MODE                  5-24-80  TSC ASSEMBLER  PAGE    1

                *
                *
                * THIS IS A UTILITY TO SET THE MODE OF AN IDS PRINTER FOR
                * CHARACTER WIDTH, AND TO TURN THE ENHANCED CHARACTER MODE ON AND
                * OFF.  SINCE THE BASIC CAN ACCESS THIS UTILITY, THE PRINTER MODE
                * MAY BE CHANGED WITHIN A BASIC PROGRAM.
                *
                * THE COMMANDS ARE AS FOLLOWS:
                *
                * CHARACTER WIDTH:
                *
                * X FOR EXTRA WIDE (8.3 CPI)
                * W FOR WIDE (10 CPI)
                * N FOR NORMAL (12 CPI)
                * S FOR SKINNY (16 CPI)
                *
                * MODE:
                *
                * E FOR ENHANCED
                * R FOR REGULAR
                * C CANCEL SPECIAL, RETURN TO 12 CPI AND NON-ENHANCED
                *
```

```
                    * NOTE THAT THIS UTIL TESTS TO SEE IF PRINT ROUTINE IS LOADED
                    * IE PRINT.SYS.  IF IT IS, P, IS NOT REQUIRED IN FRONT
                    * OF THE SET COMMAND.  IF P, IS NOT USED, AND PRINT.SYS IS NOT
                    * LOADED, THIS UTILITY EXITS TO FLEX WITH AN ERROR MESSAGE.
                    *
                    * SYNTAX: SET E      OR      SET,E
                    *
                    *
                    * FLEX2 EQUATES LIBRARY
                    *

                    *
                    *  FLEX2 EQUATES LIBRARY
                    *
 0000               PORT0   EQU    $8000
 0004               PORT1   EQU    $8004
 0008               PORT2   EQU    $8008
 000C               PORT3   EQU    $800C
 0010               PORT4   EQU    $8010
 0014               PORT5   EQU    $8014
 0014               DRVREG  EQU    $8014     DRIVE SELECT REGISTER
 0018               PORT6   EQU    $8018
 0018               CMDREG  EQU    $8018     DISK CONTROL REGISTER
 0019               TRKREG  EQU    $8019     DISK TRACK REGISTER
 001A               SECREG  EQU    $801A     DISK SECTOR REGISTER
 001B               DATREG  EQU    $801B     DISK DATA REGISTER
 001C               PORT7   EQU    $801C

 2400               LOADER  EQU    $2400     FLEX DISK LOADER ADDRESS

 A00C               PORECH  EQU    $A00C

 A840               FCB     EQU    $A840     FILE CONTROL BLOCK

                    * TTYSET CHARACTER PARAMETERS
                    * "D=$00" MEANS DEFAULT VALUE IS HEX 00
                    * "^L" MEANS CONTROL CHARACTER L
                    * "(NUL)" MEANS ASCII CONTROL CHARACTER IN THIS CASE A NULL
                    *
 AC00               BSPCHR  EQU    $AC00     BACK SPACE D=$08 ^H (BS)
 AC01               DELCHR  EQU    $AC01     DELETE CHAR D=$18 ^X (CAN)
 AC02               EOLCHR  EQU    $AC02     END OF LINE CHAR D=$3A : (COLON)
 AC03               DEPTH   EQU    $AC03     DEPTH COUNT D=0
 AC04               WIDTH   EQU    $AC04     WIDTH COUNT D=0
 AC05               NULLS   EQU    $AC05     NULL COUNT D=4
 AC06               TABCHR  EQU    $AC06     TAB CHAR D=0
 AC07               BSPECH  EQU    $AC07     BACK SPACE ECHO CHAR D=0
 AC08               EJECT   EQU    $AC08     EJECT COUNT D=0
 AC09               PAUSE   EQU    $AC09     PAUSE CONTROL D=$FF $00=NO PAUSE
 AC0A               ESCAPE  EQU    $AC0A     ESCAPE CHAR D=$1B   (ESC)
 AC0B               [illegible] EQU $AC0B    SYSTEM DRIVE NUMBER D=0
 AC0C               [illegible] EQU $AC0C    WORKING DRIVE NUMBER D=0
 AC0D               SYSTEM  EQU    $AC0D     RESERVED FOR SYSTEM
 AC0E               DATE    EQU    $AC0E     DATE REGISTER
 AC0E               MONTH   EQU    $AC0E     MONTH BYTE
 AC0F               DAY     EQU    $AC0F     DAY BYTE
 AC10               YEAR    EQU    $AC10     YEAR BYTE
 AC11               LSTTRM  EQU    $AC11     LAST TERMINATOR
 AC12               USRCMD  EQU    $AC12     USER COMMAND TABLE ADDRESS
 AC14               BUFPNT  EQU    $AC14     LINE BUFFER POINTER
 AC16               ESCRET  EQU    $AC16     ESCAPE RETURN REGISTER
 AC18               CURCHR  EQU    $AC18     CURRENT CHARACTER
 AC19               PRECHR  EQU    $AC19     PREVIOUS CHARACTER
 AC1A               CURLIN  EQU    $AC1A     CURRENT LINE NUMBER
 AC1B               LDROFST EQU    $AC1B     LOADER ADDRESS OFFSET
 AC1D               TRNSFG  EQU    $AC1D     TRANSFER FLAG
 AC1E               TRNADR  EQU    $AC1E     TRANSFER ADDRESS
 AC20               ERRNUM  EQU    $AC20     ERROR NUMBER FROM FMS

 A080               BUFFER  EQU    $A080
 A0FF               BUFEND  EQU    $A0FF

 AD00               COLDS   EQU    $AD00
 AD03               WARMS   EQU    $AD03
 AD06               RENTER  EQU    $AD06
 AD09               INCH    EQU    $AD09
 AD0C               INCH2   EQU    $AD0C
 AD0F               OUTCH   EQU    $AD0F
 AD12               OUTCH2  EQU    $AD12
 AD15               GETCHR  EQU    $AD15
 AD18               PUTCHR  EQU    $AD18
 AD1B               INBUFF  EQU    $AD1B
 AD1E               PSTRNG  EQU    $AD1E
 AD21               CLASS   EQU    $AD21
 AD24               PCRLF   EQU    $AD24
 AD27               NXTCH   EQU    $AD27
 AD2A               RSTRIO  EQU    $AD2A
 AD2D               GETFIL  EQU    $AD2D
 AD30               LOAD    EQU    $AD30
 AD33               SETEXT  EQU    $AD33
 AD36               ADDBX   EQU    $AD36
 AD39               OUTDEC  EQU    $AD39
 AD3C               OUTHEX  EQU    $AD3C
 AD3F               RPTERR  EQU    $AD3F
 AD42               GETHEX  EQU    $AD42
 AD45               OUTADR  EQU    $AD45
 AD48               INDEC   EQU    $AD48
 AD4B               DOCMND  EQU    $AD4B
                    *

 B403               FMSCLS  EQU    $B403
 B406               FMS     EQU    $B406

 BE00               READ    EQU    $BE00     READ SECTOR
 BE03               WRITE   EQU    $BE03     WRITE SECTOR
 BE06               VERIFY  EQU    $BE06     VERIFY SECTOR
 BE09               RESTOR  EQU    $BE09     RESTORE TO TRACK ZERO
 BE0C               DRSLCT  EQU    $BE0C     DRIVE SELECT
 BE0F               DREADY  EQU    $BE0F     DRIVE READY
 BE12               QREADY  EQU    $BE12     QUICK DRIVE READY
                    *
 ACE4               POUT    EQU    $ACE4
 ACC0               PINIT   EQU    $ACC0
                    *
 A100                       ORG    $A100
                    *
 A100 BD AD 27      START   JSR    NXTCH     GET COMMAND
 A103 36                    PSH A            SAVE IT
 A104 BD AD 27              JSR    NXTCH
 A107 B6 AC E4              LDA A  POUT      SEE IF PRINT.SYS LOADED
 A10A 81 39                 CMP A  #$39
 A10C 26 03                 BNE    PROC0
 A10E 7E A1 33              JMP    NOPRT
 A111 BD AC C0      PROC0   JSR    PINIT     INITIALIZE PRINTER PORT
 A114 32                    PUL A
 A115 CE A1 61              LDX    #JTAB     JUMP TABLE START
 A118 A1 00         PROC1   CMP A  0,X
 A11A 27 0A                 BEQ    FOUND1    COMMAND FOUND
 A11C 08                    INX
 A11D 08                    INX
 A11E 08                    INX
 A11F 8C A1 76              CPX    #ENDTAB
 A122 27 06                 BEQ    NFERR     NOT FOUND
 A124 20 F2                 BRA    PROC1
 A126 EE 01         FOUND1  LDX    1,X       GET START ADDRESS FOR COMMAND
 A128 6E 00                 JMP    0,X       JUMP THERE
                    *
                    * ERROR ROUTINES
                    *
 A12A CE A1 76      NFERR   LDX    #NFSTR    NOT FOUND
 A12D BD AD 1E              JSR    PSTRNG
 A130 7E AD 03              JMP    WARMS     EXIT TO FLEX
                    *
 A133 CE A1 89      NOPRT   LDX    #NPRINT   PRINT ROUTINE NOT LOADED
 A136 BD AD 1E              JSR    PSTRNG
 A139 7E AD 03              JMP    WARMS     EXIT
                    *
                    * SET ROUTINES
                    *
 A13C 86 1C         SETX    LDA A  #$1C      8.3 CPI
 A13E 20 1D                 BRA    OUT
                    *
 A140 86 1D         SETW    LDA A  #$1D      10 CPI
 A142 20 19                 BRA    OUT
                    *
 A144 86 1E         SETN    LDA A  #$1E      12 CPI
 A146 20 15                 BRA    OUT
                    *
 A148 86 1F         SETS    LDA A  #$1F      16 CPI
 A14A 20 0F                 BRA    OUT
                    *
 A14C 86 01         SETE    LDA A  #$01      EXPANDED OR DOUBLE WIDE
 A14E 20 0B                 BRA    OUT
                    *
 A150 86 02         SETR    LDA A  #$02      REGULAR OR NORMAL WIDTH
 A152 20 07                 BRA    OUT
                    *
 A154 86 02         SETC    LDA A  #$02      SET NORMAL WIDTH
 A156 BD AC E4              JSR    POUT
 A159 86 1E                 LDA A  #$1E      SET 12 CPI
                    *
 A15B BD AC E4      OUT     JSR    POUT      OUTPUT TO PRINTER
 A15E 7E AD 03              JMP    WARMS     RETURN TO FLEX
                    *
                    * JUMP TABLE
                    *
 A161               JTAB    EQU    *
                    *
 A161 58                    FCB    'X
 A162 A1 3C                 FDB    SETX
 A164 57                    FCB    'W
 A165 A1 40                 FDB    SETW
 A167 4E                    FCB    'N
 A168 A1 44                 FDB    SETN
 A16A 53                    FCB    'S
 A16B A1 48                 FDB    SETS
 A16D 45                    FCB    'E
 A16E A1 4C                 FDB    SETE
 A170 52                    FCB    'R
 A171 A1 50                 FDB    SETR
 A173 43                    FCB    'C
 A174 A1 54                 FDB    SETC
 A176               ENDTAB  EQU    *
                    *
                    * STRINGS
                    *
 A176 43            NFSTR   FCC    /COMMAND NOT FOUND /
 A188 04                    FCB    4
 A189 50            NPRINT  FCC    /PRINT.SYS NOT LOADED /
 A19E 04                    FCB    4
                            END    START

NO ERROR(S) DETECTED
```

```
SET PRINTER IN A/BASIC                              5-24-80        PAGE 1

        REM         SETBAS SET PRINTER IN A/BASIC MUST USE P, PRECEEDING
        REM         COMMAND

            OPT S
            BASE=$30

10      REM         POKE SWITCH FOR PROMPT TO TERMINAL
            POKE(-21470)=-1

            PRINT
            PRINT "PRINTER MODE";
            INPUT M$

        REM         POKE SWITCH FOR OUTPUT TO PRINTER
            POKE(-21470)=0

            IF M$="X" THEN 51
              IF M$="W" THEN 52
                IF M$="N" THEN 53
                  IF M$="S" THEN 54
                    IF M$="E" THEN 55
                      IF M$="R" THEN 56
                        IF M$="C" THEN 57

                           PRINT "ILLEGAL COMMAND" : GOTO 10

51                        PRINT CHR$(28) : GOTO 80
52                      PRINT CHR$(29) : GOTO 80
```

```
53              PRINT CHR$(30) : GOTO 80
54            PRINT CHR$(31) : GOTO 80
55          PRINT CHR$(1) : GOTO 80
56        PRINT CHR$(2) : GOTO 80
57      PRINT CHR$(2)+CHR$(30) : GOTO 80

80      END
        STOP
```

```
ALTERNATE SET PRINTER IN A/BASIC              5-24-80        PAGE 1

     REM        SETBAS SET PRINTER IN A/BASIC. MUST USE P. PRECEEDING
     REM        THE COMMAND.

          BASE=$30

10   REM        POKE SWITCH FOR PROMPT TO TERMINAL
          POKE (-21479)=-1

          PRINT
          PRINT "PRINTER MODE";
          INPUT M$

     REM        POKE SWITCH FOR OUTPUT TO PRINTER
          POKE (-21470)=0

          IF M$<>"X" THEN 20
            PRINT CHR$(28) : GOTO 90

20        IF M$<>"W" THEN 30
            PRINT CHR$(29) : GOTO 90

30        IF M$<>"N" THEN 40
            PRINT CHR$(30) : GOTO 90

40        IF M$<>"S" THEN 50 : GOTO 90
            PRINT CHR$(31) : GOTO 90

50        IF M$<>"E" THEN 60
            PRINT CHR$(1) : GOTO 90

60        IF M$<>"R" THEN 70
            PRINT CHR$(2) : GOTO 90

70        IF M$<>"C" THEN 80
            PRINT CHR$(2)+CHR$(30) : GOTO 90

80        PRINT "ILLEGAL COMMAND"
          GOTO 10

90        STOP
          END
```

```
SET PRINTER IN PASCAL                         5-24-80        PAGE 1

PROGRAM SETPRINT;

(* ENTER WITH P. PRECEEDING NAME *)

CONST
   EXTRAWIDE = $1C;
   WIDE      = $1D;
   NORMAL    = $1E;
   SKINNY    = $1F;
   ENHANCED= 1;
   REGULAR = 2;
   SWITCH = $AC22;
   ON     = $FF;
   OFF    = 0;

VAR
   MODE : CHAR;

BEGIN
   POKE (SWITCH,ON); (* SWITCH OUTPUT TO TERMINAL *)
   WRITE ('PRINTER MODE ? ');
   READ (MODE);
   WRITELN;
   POKE (SWITCH,OFF);

   CASE MODE OF

    "X": WRITE (CHR(EXTRAWIDE));
    "W": WRITE (CHR(WIDE));
    "N": WRITE (CHR(NORMAL));
    "S": WRITE (CHR(SKINNY));
    "E": WRITE (CHR(ENHANCED));
    "R": WRITE (CHR(REGULAR));
    "C": WRITE (CHR(REGULAR), CHR(NORMAL));
   END; (* CASE MODE OF *)

END. (* SETPRINT *)
```

```
TEST OF SET FROM BASIC                        5-24-80        PAGE 1

10 OPEN "O.PRINT" AS 0
20 PRINT #0
30 PRINT #0, "NORMAL CONDITION"
40 EXEC, "SET S"
50 PRINT #0,"MOST NARROW PRINTING, 16 CPI"
60 EXEC, "SET W"
70 PRINT #0,"WIDE NORMAL AT 10 CPI"
80 EXEC, "SET X"
90 PRINT #0,"WIDEST PRINTING 8.3 CPI"
100 EXEC, "SET S"
110 EXEC, "SET E"
120 PRINT #0,"16 CPI DOUBLE WIDE"
130 EXEC, "SET N"
140 PRINT #0,"12 CPI DOUBLE WIDE"
150 EXEC, "SET W"
160 PRINT #0,"10 CPI DOUBLE WIDE"
170 EXEC, "SET X"
180 PRINT #0,"8.3 CPI DOUBLE WIDE"
190 EXEC, "SET C" : REM RESTORE TO NORMAL
200 PRINT #0
210 CLOSE 0
220 END
```

```
NORMAL CONDITION
MOST NARROW PRINTING, 16 CPI
WIDE NORMAL AT 10 CPI
WIDEST PRINTING 8.3 CPI
16 CPI DOUBLE WIDE
12 CPI DOUBLE WIDE
10 CPI DOUBLE WIDE
8.3 CPI DOUBLE WIDE
```

HUMBUG

HUMBUG - A NEW 6800 MONITOR

by Dale L. Puckett

This month we look at HUMBUG, a new monitor which should prove to be a lot of help to the 6800 user programming in assembly or machine language without the benefit of a disk system.

HUMBUG was written by Pete Stark and is available from STAR-KITS, P. O. Box 209, Mt. Kisco, New York, 10549. There are 2K, 3K and 4K versions burned in 2708 or 2716 EPROMs. All versions come with a complete source listing.

The 2K version in a 2716 sells for $40. The 4K version is $65. in the 2708 form HUMBUG will cost you $45 for the 2K version, $60 for 3K of code and $75 for 4K.

The source code listing is available on FLEX 2.0, mini-FLEX and Percom disks, and on paper. The listing costs $20. The source on disk costs $25.

The 2716 versions were designed to work on the SWTPC MP-A2 CPU board. If you buy the 2708 version, you must supply a separate EPROM board.

COMPATIBILITY

HUMBUG maintains all of SWTBUGs major subroutine entry points. This allows all of your old programs to run without change. Also, the SWTBUG scratchpad addresses were left intact.

HUMBUG gives you full control over the system from your keyboard. You can even turn various ports on and off during the execution of a program. Another feature allows you to abort programs from the keyboard without pushing the RESET button.

Additionally, Stark has added extended debugging facilities. These include an impressive handling of multiple breakpoints and a special single stepping mode.

MIKBUG and SWTBUG both used one stack. HUMBUG uses two. By keeping the user and monitor stacks completely separate, Stark has insured that the two will never clobber each other.

I/O SCHEME

HUMBUG uses port 1 for all monitor input and most output. An ACIA such as the SWTPC MP-S or Star-Kits CT-PS card must be installed. This monitor uses an address called PORADD to allow you to redirect the control port by simply changing the $8004 at PORADD to the address of a port where you have installed a serial interface card.

It is also possible for HUMBUG to output to a second serial card installed at port 0 or to special

output routines like those that drive my IBM selectric. HUMBUG continuously checks the control port for an ASCII DC3 or XOFF character (control-C). When it receives this character it echoes a BELL and stops all output. It then inputs one more character to check for any port change commands.

Options include: "0," which toggles port 0 on and off; 1, which does the same for port 1; D, which turns the special user written port routine on and off; and P, which turns the pause mode on and off. This mode is used for video terminals and stops the output automatically after every 16 lines.

If a carriage return is received immediately after the XOFF the program is aborted and control returns to HUMBUG. Unless, that is, the program has changed the address in HUMBUG's return address pointer to send control back to itself. Stark didn't miss a thing.

COMMANDS

Because of the excellent prompting this monitor is almost foolproof. For example if you type AD which is HUMBUG's command for a formatted ASCII dump, it will ask you FROM and wait for you enter an address. Then, it will ask you, TO. After you type this second address your dump will begin. And check this, if you are working with the same block of addresses you may simply type a carriage return instead of the FROM address.

The AD command prints the starting address of the dump at the beginning of a line followed by the ASCII characters for the next 16 bytes. If a location has a value of less that $1F, a period '.' is printed. Also, the parity bit is ignored by this dump routine.

Let's look at the rest of HUMBUG's commands now. They are in alphabetical order in the fine users manual supplied by STAR-KITS. We will leave them that way here.

"AI" is the command to input an ASCII string. It prompts you for a FROM/TO pair after which you simply type in ASCII characters. If you type more text than you have allowed space for when you answered the FROM/TO prompt, the routine will begin printing ERROR on the screen until you type XOFF.

If you run out of text before you run out of allotted space, you may simply type XOFF/CR. When you do this, HUMBUG will automatically change the FROM/TO pair so that if you wanted a dump of the area to check the accuracy of your typing you could simply answer the FROM prompt with a carriage return and you will automatically dump the correct portion of memory.

"AO" stands for ASCII output and allows you to output the contents of memory to port 1. It uses the FROM/TO prompt and is very useful for copying tapes, etc.

Typing "BP" will give you a listing of the four breakpoints that you have set. It gives you the address and the opcode or instruction normally stored at each breakpoint.

"BR" is the command that allows you to set breakpoints. It prompts you for the number of the breakpoint you wish to set or change and the address you desire to change it to. If you enter a CR to answer the address prompt the breakpoint will be removed. If you enter a valid address, that address will become a breakpoint.

"CL" will clear the standard SWTPC terminal screen. It outputs a $10, $16 string. "CO" allows you continue execution of a program after reaching a breakpoint. It also allows you continue at full speed after a sequence of single step operations. You must however, remove the breakpoint before continuing or you will not be able to proceed.

Typing "CS" will print a 16 bit checksum of the area within the FROM/TO range. You are prompted for the FROM/TO values.

"DE" is the command for a mini-disassembler. It prints a formatted dump with the memory contents formatted into one, two or three byte groups. In other words, if it encounters a CLR A instruction, only "$4F" will be printed on that line. If it hits a JMP MIKBUG, it would print "7E E0D0" on one line, etc. It does not print the mnemonic value of the instructions.

"EN" outputs the standard end-of-tape code. This includes the contents of the program counter followed by an "S9".

"FA" is a fastype mode. It generates a tape which may be used for input to an incompatible editor or BASIC interpreter, etc. The carriage return is followed with a one-half second pause and all other control commands are eliminated from the data stream.

"FD" is the command you would use to boot a FLEX disk. "FI" is a short search routine which allows you to find one, two or three byte values. It prompts for the number of bytes you wish to search for, the value of those bytes and the FROM/TO pair which will tell the routine where to look.

"FM" allows you to fill memory. It prompts you for the area to fill with the standard FROM/TO prompt and prompts you for the value of the fill byte. One excellent use of this command is to fill all unused memory with SWI commands, "3 ". If you do this and your program bombs, the resulting register dump will tell you where it went astray.

"GO" will always jump to the address stored at $A048. "GA" will print out the contents (address) stored at $A048. It merely prints the program counter and shows you where the computer would start execution if you typed "GO."

"HD" will print a standard hexadecimal dump of memory between FROM and TO. The address appears at the beginning of each line followed by sixteen bytes.

"HE" stands for Help! It will print a summary of HUMBUG commands. It is only installed in the 2K versions.

"IQ" enables an interrupt routine so that the serial printer on port 0 can be used with interrupts to fill and print from a 1K buffer. This allows you to print and compute at the same time. It is available in the 4K version only.

"JU" is similar to the "J" command in SWTBUG. However, it uses a JSR instruction instead of a JMP instruction. This means that if there is an RTS at the end of the users program control will return to HUMBUG. It gives you an easy method of testing subroutines.

"LO" will load a tape on the control terminal. "MC" will compare the contents of two areas of memory. It prints out the memory contents for each byte that is different. It prompts you for the FROM/TO combination for the first memory area and the FROM address of the second. This particular command's output is one of the only HUMBUG formats I didn't like. It dumps its output continuously without the benefit of any carriage return/line feed pairs and is hard to read.

"ME" stands for memory examine and change and works exactly like SWTBUG's M command.

"MO" is the move memory command. The contents of memory in an area you name by answering the FROM/TO prompts is moved to another area. It should be noted that this routine even allows you to move data where part of the new area overlaps the original area. This is a feature you don't see too often.

"MT" is a simple rotating bit memory test. It checks the area within the FROM/TO area and prints a "+" sign if everything is ok. If there is a memory problem, it prints the address of the bad location and the contents of that location at the time the test failed.

"PU" is similar to SWTBUG's "P" command except that it prompts for the addresses you desire to punch.

"RE" which stands for register examine is a step ahead of SWTBUG's "R" also. The main improvement is in the format of the printing of the value of the CC register. HUMBUG prints it in binary which allows you to visualize which bits are set much easier. Also the printed stack pointer address tells you where the stack was pointing just before the breakpoint, rather than where it was pointing after returning to the monitor. It makes a whole lot of sense.

"SS" stands for single step. It prints the register contents after executing each intruction. It must however follow the "ST" or start single stepping command. "ST" prompts for the address where you want to start single stepping. "SS" prints out two lines. The first shows you the address and code of the instruction which you are checking. The second line is a register dump after the instruction has been executed. It is really a help for spotting logic errors in your code.

Other commands include "ZZ" which is a simple jump to $C000, the address of Percom's DOS, and 30 or 12. The latter pair allows you to change your apparent baud rate by changing the initialization on the ACIA. If you change from a 16 times clock to a 64 times clock you will slow down your output baud rate by 4. I must be honest and say that this particular feature did not work for me when I tried it. I am sure it has something to do with the way I have my hardware configured but I just don't have the time to pursue it.

MEMORY REQUIREMENTS

The 2K versions only use the 128-byte scratchpad RAM at $A000 on the MP-A or MP-A2 CPU cards. The 3K version requires that you have another scratchpad RAM at $D000-$D07F. The instruction manual shows you a very simple and inexpensive way to implement this additional memory.

The 4K versions require an additional 1K buffer at $D400 for use as a buffer for the interrupt driven printer. Again, the users manual tells you how to set it all up. It even shows you how to convert an old 4K memory board for use at $D400.

CONCLUSION

HUMBUG is one of the better monitors to come down the pike. The built in debugging utilities would have helped me a great deal several years ago when I was struggling to learn this business without the benefit of a disk system.

It is so well prompted that you could almost use it without reading the users manual. Once you know the two letter codes you are home free.

The manual itself is attractively packaged and contains all the information you will ever need to exercise HUMBUG. The source code provided is extremely well commented and is an educational tool itself. I would highly recommend that anyone interested in talking to the 6800 right down at the gut level use this tool to speed them up the learning curve. I only wish it had been available three years ago.

Note: The problem mentioned above under the Command 'LO' is apparent only on narrow CRT lines (32 or 64 characters per line). CRT displays that exhibit 80 characters per line do honor the CR/LF and are more readable.

MC6883

The Motorola MC6883 is the 'heart' of the Radio Shack TRS80C™ Color Computer, that is being delivered now. It is a new device and little has been published on it so far. It uses a 6809E and other Motorola devices - unfortunaly no S50 bus. The following technical data is presented due to the large amount of interest indicated by the letters and calls we have received. This is poorly reproduced; we could do no better. We run it hoping that with a little quessing and magnifying glass operation you may garner some interesting facts and values. We trust that this machine will allow the beginning user to realize what a fine device the 6809 is; and then upgrade to a 'real' 6809 computer - just like yours and mine!

MC 6883
(74LS783)

SYNCHRONOUS ADDRESS MULTIPLEXER

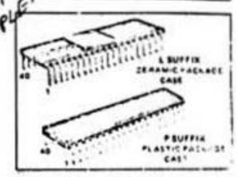
(LOW POWER SHOTTKY)

Advance Information

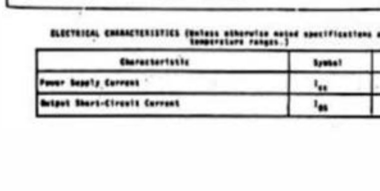
The Motorola MC6883 brings together the MC6809E (MPU), the MC6847 (Color Video Display Generator) and Dynamic RAM to form a highly effective, compact and cost effective computer and display system.

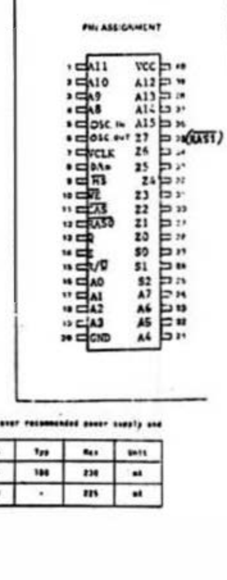
- MC6809E and MC6847 (VDG) compatible
- Transparent MPU/VDG/Refresh
- RAM size: 4K, 8K, 16K, 32K or 64K bytes (dynamic or static)
- Addressing range: 96K bytes
- Single crystal provides all timing
- Register programmable:
  - VDG addressing modes
  - VDG offset (0 to 64K)
  - RAM size
  - Bank switch
  - MPU rate (crystal ÷ 16 or ÷8)
  - MPU rate...address dependent or independent
- System device selects decoded 'on chip'
- Timing is optimized for standard RAMs
- +5V only operation
- Easy synchronization of multiple SAM systems
- DMA mode

DATA SHEET NOT COMPLETE YET!



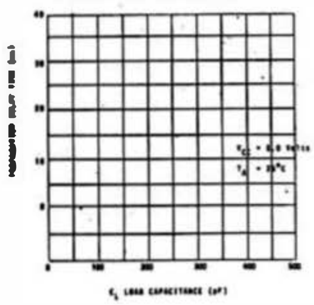
SYSTEM BLOCK DIAGRAM



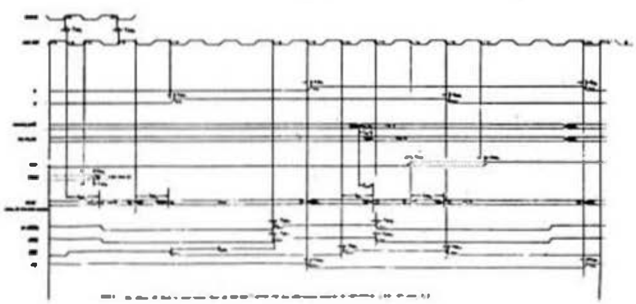


ELECTRICAL CHARACTERISTICS (Unless otherwise noted specifications apply over recommended power supply and temperature ranges.)

Characteristic	Symbol	Min	Typ	Max	Unit
Power Supply Current	I_{CC}	-	100	230	mA
Output Short-Circuit Current	I_{OS}	30	-	225	mA



1C = 64x64
2C = 128x64
3C = 128,96
6C = 128,192
6R = 256,192



VDC ADDRESS OFFSET

Seven bits (F_6, F_5, F_4, F_3, F_2, F_1 and F_0) determine the "STARTING ADDRESS" for the video display. The "Starting Address" is defined as "the address corresponding to data displayed in the UPPER LEFT corner of the T.V. screen". Thus, the "Starting Address" is shown below in binary:

F_6	F_5	F_4	F_3	F_2	F_1	F_0	0	0	0	0	0	0	0	0	0

Most Significant Bit — Least Significant Bit

Note that the "Starting Address" may be placed anywhere within the 64K address space with a resolution of 1K (the size of one alphanumeric page).

The F_6-F_0 bits take effect during the T.V. vertical synchronization pulse (i.e., when FS from MC6847 is low).

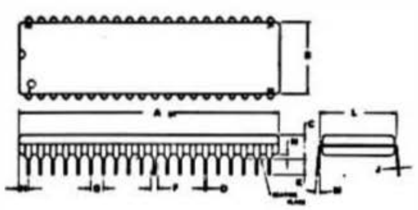
PAGE SWITCH

One bit (P1) is used "in place of" A15 from the 6809 in order to refer access within \$0000-\$7FFF to one of two 32K byte pages of RAM. If the system does not use more than 32K bytes, P1 can be ignored.*

*When using 4K x 1 RAMS, two banks of eight IC's are allowed. This accounts for Addresses \$0000-1FFF. Also, this same RAM can be addressed at \$2000-\$3FFF, \$4000-\$5FFF and \$6000-\$7FFF.

	MODE TYPE	MC6847 MODE					S.A.M. MODE			
		$\bar{A}/G$	GM2	GM1	GM0 & $\bar{I}$/EXT	CSS	V_2	V_1	V_0	
1	INTERNAL ALPHANUMERICS	0	X	X	0	X	0	0	0	
2	EXTERNAL ALPHANUMERICS	0	X	X	1	X	0	0	0	
3	SEMIGRAPHICS - 4	0	X	X	0	X	0	0	0	
4	SEMIGRAPHICS - 6	0	X	X	1	X	0	0	0	
5	FULL GRAPHICS- 1C	1	0	0	0	X	0	0	1	64,64
6	FULL GRAPHICS- 1R	1	0	0	1	X	0	0	1	128,64 B/W
7	FULL GRAPHICS- 2C	1	0	1	0	X	0	1	0	128,64
8	FULL GRAPHICS- 2R	1	0	1	1	X	0	1	1	128,96
9	FULL GRAPHICS- 3C	1	1	0	0	X	1	0	0	
10	FULL GRAPHICS- 3R	1	1	0	1	X	1	0	1	
11	FULL GRAPHICS- 6C	1	1	1	0	X	1	1	0	
12	FULL GRAPHICS- 6R	1	1	1	1	X	1	1	0	
13	DIRECT MEMORY ACCESS	X	X	X	X	X	1	1	1	

P SUFFIX
PLASTIC PACKAGE
CASE 711-02

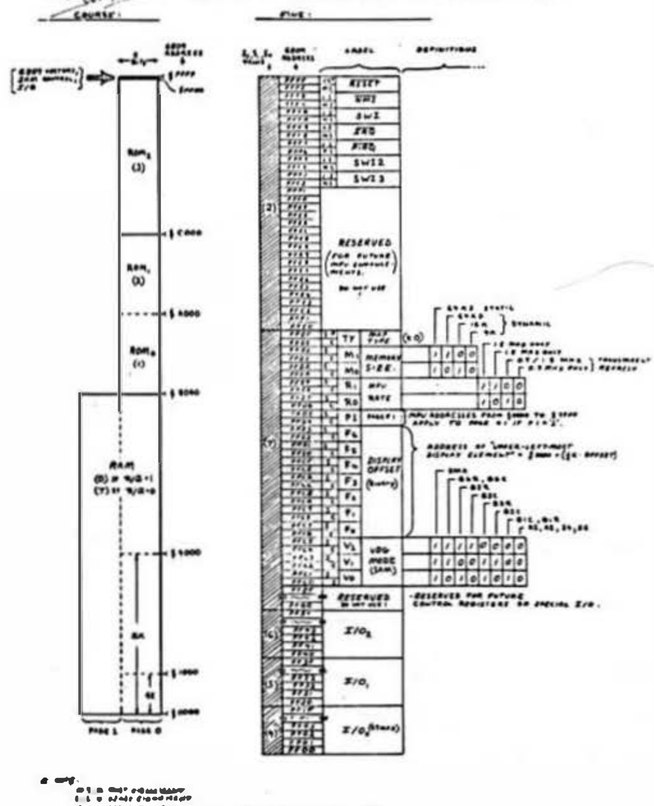


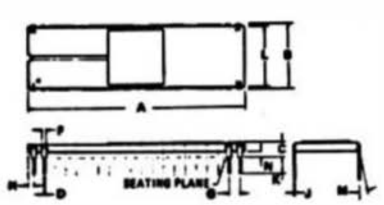
DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

CASE 711-02

Color Computer

SAM COMPUTER MEMORY MAP



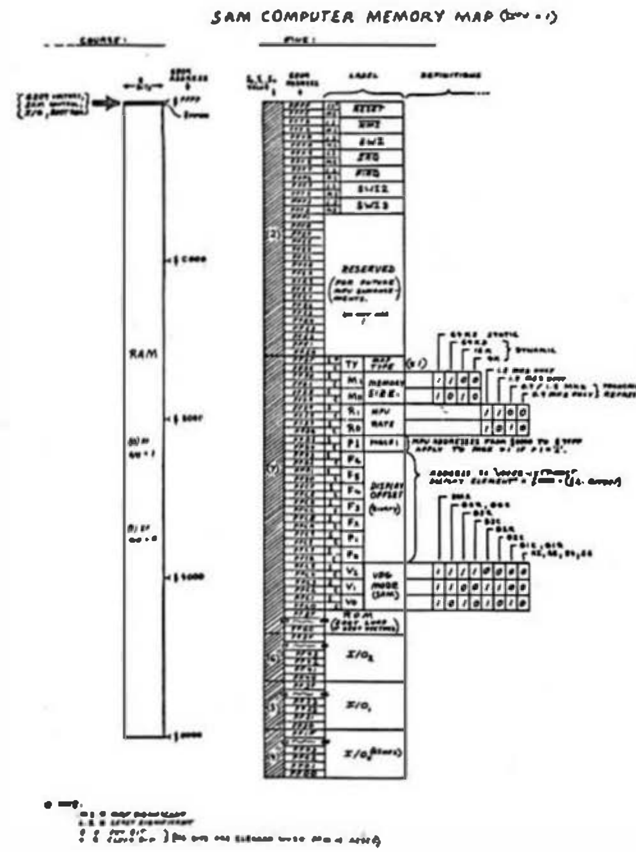


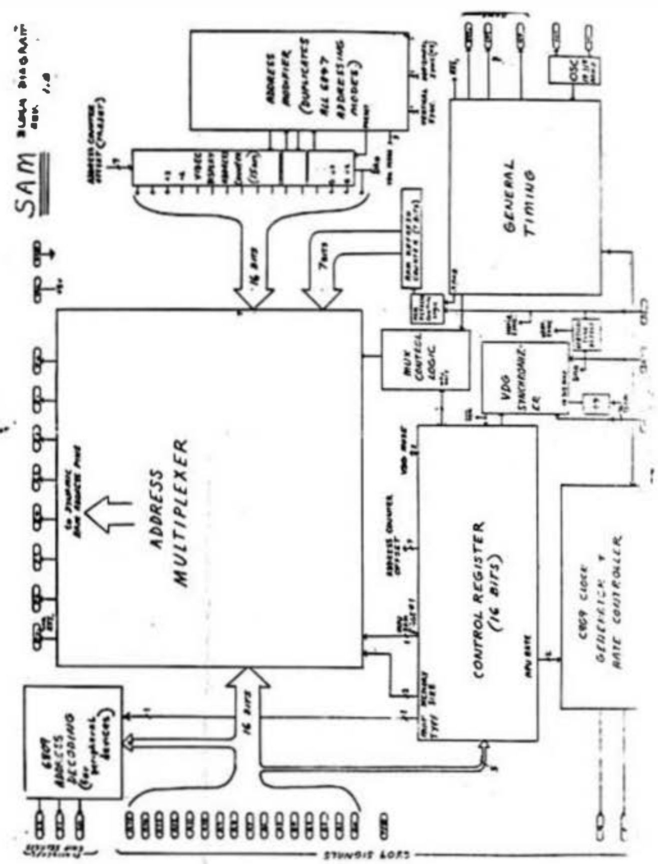
DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	50.29	51.31	1.980	2.020
B	14.05	15.62	0.553	0.615
C	2.54	4.19	0.100	0.165
D	0.38	0.53	0.015	0.021
F	0.76	1.40	0.030	0.055
G	2.54 BSC		0.100 BSC	
H	0.76	1.78	0.030	0.070
J	0.20	0.33	0.008	0.013
K	2.54	4.19	0.100	0.165
L	14.60	15.37	0.575	0.605
M	0°	10°	0°	10°
N	0.51	1.52	0.020	0.060

CASE 715-02

L SUFFIX
CERAMIC PACKAGE
CASE 715-02

NOTE:
1. LEADS, TRUE POSITIONED WITHIN 0.25 mm (0.010) DIA (AT SEATING PLANE), AT MAX. MAT'L CONDITION.





MC6883 / 74LS783 (SAM) PRELIMINARY INFORMATION

ABSOLUTE MAXIMUM RATINGS

(ABOVE WHICH THE USEFUL LIFE MAY BE IMPAIRED)

Storage Temperature	-65°C to +150°C
Temperature (Ambient) Under Bias	0°C to +70°C
Vcc Pin Potential to Ground Pin	-0.5 V to +7.0 V
*Input Voltage (dc) Diode Inputs	-0.5 V to 10 V
*Input Current (dc)	-30mA to +5.0 mA
Voltage Applied to Outputs (Output High)	-0.5 V to + 7 V

*Either input voltage limit or input current limit is sufficient to protect the inputs.

GUARANTEED OPERATING RANGES

Supply Voltage (V_{CC}) Min	Typ	Max	Temperature
4.75 V	5.0 V	5.25 V	0°C to +70°C

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

Symbol	Parameter	Limits Min	Typ	Max	Units	Test Conditions
V_{IH}	Input HIGH Voltage	2.0			V	ALL INPUTS
V_{IL}	Input LOW Voltage			0.8	V	ALL INPUTS
V_{CD}	Input Clamp Diode Voltage			-1.5	V	V_{CC} = MIN I_{IN} = -18 mA
I_{IH}	Input HIGH Current ALL INPUTS			20	μA	V_{CC} = MAX V_{IN} = 2.7 V
	VCLK INPUT			[illegible]	mA	V_{CC} = MAX V_{IN} = [illegible]
	ALL OTHER INPUTS			[illegible]	mA	V_{CC} = MAX V_{IN} = 7 V
I_{IL}	Input LOW Current VCLK INPUT			[illegible]	mA	V_{CC} = MAX
	ALL OTHER INPUTS			-0.4		V_{IN} = 0.4V
V_{OH}	Output HIGH Voltage [illegible]	[illegible]			V	Ioh = -1.0mA, Vcc = MIN
V_{OH}	[illegible]	[illegible]			V	I_{OH} = -0.2 mA
V_{OH}	ALL OTHER OUTPUTS	[illegible]			V	I_{OH} = [illegible], Vcc = MIN
V_{OL}	Output LOW Voltage [illegible]			[illegible]	V	I_{OL} = [illegible], Vcc = MIN
V_{OL}	VCLK			[illegible]	V	I_{OL} = [illegible], Vcc = MIN
V_{OL}	ALL OTHER OUTPUTS			[illegible]	V	I_{OL} = [illegible], Vcc = MIN

MC6883 / 74LS783 (SAM) PRELIMINARY INFORMATION

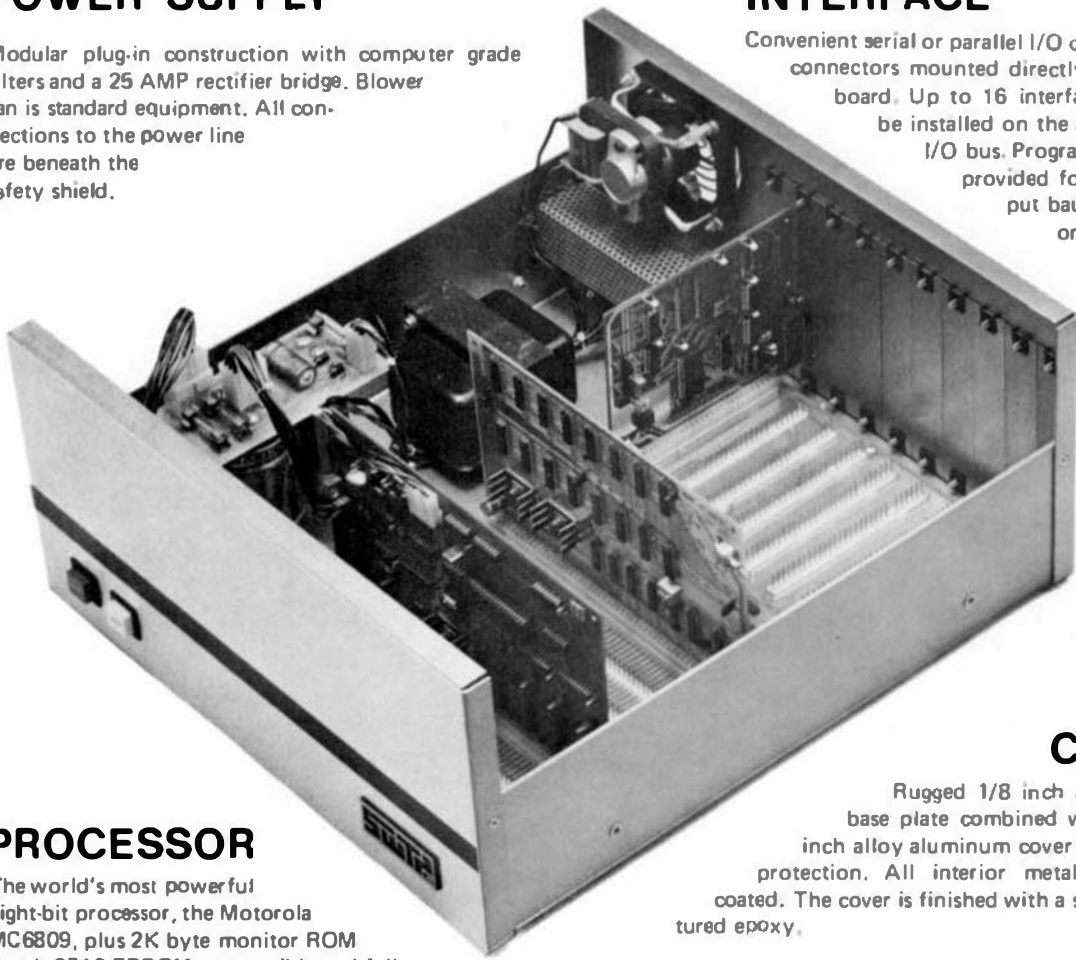
AC CHARACTERISTICS

UNLESS OTHERWISE NOTED [illegible]

Symbol	Parameter	Min	Typ	Max	Units	Test Conditions
[illegible]	[illegible]				ns	[illegible]

NOTE "1": [illegible]

MOTOROLA SEMICONDUCTOR PRODUCTS INC.



6809 DISK SYSTEMS

All disk systems are supplied with our version of FLEX 9, the world standard disk operating system for the 6809. Our systems normally operate in double density format, but they are compatible with single density, or single sided recording formats. FLEX is supplied with over forty utilities, many of which are only available with our systems.

Our disk systems offer you mass storage at low cost. The cost per thousand bytes of storage for our various systems is shown in the chart. Other 6809 disk systems have costs up to three times greater for the same general type drive.

TYPE	CAPACITY	COST
D-5	720,000 bytes	$1.80 per/K
DT-5	1,400,000 bytes	$1.16 per/K
DMF-2	2,400,000 bytes	$1.04 per/K
CDS-1	16,000,000 bytes	$ .27 per/K

D-5 Two double sided, double density, 5" disk drives with a total on line capacity of 720,000 bytes of data. Includes cabinet, power supply, connecting cable and controller. Controller will operate up to four drives. This is an ideal disk system for small stand alone word processing systems, or for businesses that do not work with large inventories.

14 x 6 x 10 — 20 lbs $1,295.00

D-5 or DT-5

DT-5 Double track density version of the D-5. The DT-5 uses two 96 track per inch drives to provide an on line capacity of 1,400,000 bytes. Includes cabinet, power supply, connecting cable and controller. Controller will operate up to four drives. This is a disk system with enough capacity to include small inventories of up to 1,000 items, plus the usual business package of general ledger payroll, etc.

14 x 6 x 10 — 20 lbs $1,695.00

DMF2

DMF-2 Double sided, double density, dual eight-inch disk system with an on line capacity of 2,400,000 bytes. Our "top of the line" disk system features a DMA type controller for fastest possible data transfers. This drive was designed for larger businesses and multi user installations. The DMF-2 will provide the fast operation necessary for systems running multiterminals under the UniFLEX operating system. Complete with a heavy duty 1/8-inch metal cabinet, power supply, connecting cable and controller. The controller will operate up to four drives.

17½ x 5 x 21½ — 53 lbs $2,495.00

CDS-1 This "Winchester" type hard disk provides both large storage capacity and high speed operation. The CDS-1 is the answer for systems that must handle large inventories or systems with more than four terminals. The controller has its own processor and uses DMA data transfer.

CDS-1 — 115 lbs $4,395.00

CDS-1

SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

S.A.M. (SYNCHRONOUS ADDRESS MULTIPLEXER)

PROGRAMMING GUIDE

S.A.M. - GENERAL DESCRIPTION

The S.A.M. is a 40 pin integrated circuit to be used with the MC6809 microprocessor unit (MPU), the MC6847 color Video Display Generator (VDG) and up to 64K bytes of dynamic RAM.

As the name implies, the S.A.M. multiplexes the MC6809 addresses and the MC6847 addresses into eight dynamic RAM address pins. In addition, the S.A.M. provides complete system address decoding and timing (including $\overline{RAS}$, $\overline{CAS}$ and $\overline{WE}$). The S.A.M. provides clocks to the MC6809, MC6847 and the internal refresh counter in such a manner as to SYNCHRONIZE all three.

S.A.M. - PROGRAMMABILITY

The S.A.M. contains a 16 bit control register which allows the 6809 to program the S.A.M. for the following options:

VDG ADDRESSING MODE - 3 bits
VDG ADDRESS OFFSET -- 7 bits
32K PAGE SWITCH ----- 1 bit
MPU RATE ------------ 2 bits
MEMORY SIZE --------- 2 bits
MAP TYPE ------------ 1 bit

Note that when the S.A.M. is reset by first applying power or by manual hardware reset, all control register bits are cleared (to a logic "0").

VDG ADDRESSING MODE

Three bits (V_2, V_1, V_0) control the sequence of DISPLAY ADDRESSES generated by the S.A.M. (which are used to scan dynamic RAM for video information). For example, if you wish to display Dynamic RAM data as INTERNAL ALPHANUMERICS VIDEO, you should program* the MC6847 for the INTERNAL ALPHANUMERICS MODE and CLEAR BITS V_2, V_1 and V_0 in the S.A.M. The table on the following page summarizes the available modes:

*In a "Sam computer" , a PIA at location $FF22 is used to control MC6847 modes. (See MC6847 Data Sheet.)

MPU RATE

Two bits (R_1, R_0) control the clock rate to the MC6809 MPU. The options are:

RATE (FREQUENCY OF "E" CLOCK)	R_1	R_0
0.9 MHz (CRYSTAL FREQUENCY ÷ 16)	0	0
0.9/1.8 MHz (ADDRESS DEPENDENT RATE)	0	1
1.8 MHz (CRYSTAL FREQUENCY ÷ 8)	1	X

(Typical Crystal Frequency = 14.31818 MHz)

In the "address dependent rate" mode, accesses to $0000-$7FFF and $FF00-$FF1F are slowed to 0.9 MHz and all other addresses are accessed at 1.8 MHz.

MEMORY SIZE*

Two bits (M_1 and M_0) determine RAM memory size. The options are:

SIZE	M_1	M_0
One or two banks of 4K x 1 dynamic RAMS	0	0
One or two banks of 16K x 1 dynamic RAMS	0	1
One bank of 64K x 1 dynamic RAMS	1	0
Up to 64K static RAM**	1	1

IMPORTANT!! Be sure to program the S.A.M. for the correct memory size before using RAM (i.e., for a subroutine stack).

MAP TYPE*

One bit (TY) is used to select between two memory map configuarations. The options are:

TYPE	T_Y
RAM: $0000-7FFF ROM: 8000-FEFF I/O, VECTORS, SAM CONTROL REG.: $FF00-$FFFF	0
RAM: $0000-FEFF I/O, VECTORS, ROM, SAM CONTROL REG.: $FF00-$FFFF	1

* Refer to Appendices A and B.

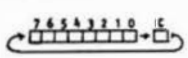
** Requires an extra latch for demultiplexing the RAM address.

WRITING TO THE S.A.M. CONTROL REGISTER

Any bit in the control register (C.R.) may be set by writing to a specific unique address. Each bit has two unique addresses...writing to the even # address clears the bit and writing to the odd # address sets the bit. (Data on the data bus is irrelevant in this procedure.) The specific addresses are tabulated in Appendix A and Appendix B.

If desired, a short routine may be written to program the S.A.M. C.R. "a word at a time". For example, the following routine copies "B" bits from "A" register to S.A.M. C.R. addresses beginning with address "X".

```
SAM1   46       ROR A
       24  06   BCC SAM2
       30  01   INX (LEAX1,X)
       A7  00   STA 0,X*
       20  02   BRA SAM3
SAM2   A7  81   STA 0,X**
SAM3   5A       DEC B
       26  F2   BNE SAM1
       39       RTS
```



RMS dbms

Larry Kherlaty, Washington Computer Services

DBMS: WHAT AND WH ?

Everybody knows that computers can be programmed in BASIC or PASCAL or whatever to do all sorts of record keeping functions cheap. Right...? Wrong! We all watch in delight as computer power increases, and computer prices decrease. But, even though the hardware is ready for business, the software is not. The reason is simply that software is a labor-intensive product. And to make matters worse, the software laborer must be highly skilled, (and highly paid). Even the most efficient programmer is forced to wade through a forest of messy details just to get to the point where he can attack the particular problem at hand. Many a business system has been developed by brute force only to find that to alter it for day-to-day changes in the real world, or to modify it to fit another business, can be accomplished only by investing further amounts of costly programmer time. The solution to this problem lies in the application of the right software tools. RMS, a database management system, is such a tool.

The theory behind database management is that there is a fairly broad class of record keeping problems which all share certain common aspects. These common aspects are handled in a general way. Thus, when attacking a particular data handling problem, many of the tedious programming tasks that would otherwise need to be done just to get off the ground, are taken care of automatically by the DBMS. The common aspects of any computerized record keeping system are: 1) putting individual pieces of information into the database; 2) modifying the information in the database; 3) fetching information from the database on command; 4) re-arranging the information in the database, and, 5) producing printed reports from the database. As you know, all of these things can be accomplished by writing some programs to manipulate files. However, a set of such programs is NOT a database management system. With a database management system it is possible to perform all of these tasks without ever writing any programs. The user need only describe the data items that are to be managed; the DBMS takes care of the details. Record Management System (RMS) is a set of extremely efficient 6809 machine language programs that make up a versatile and understandable DBMS.

THE DATA DICTIONARY

To establish a database using RMS, the first step is to create a text file called a DICTIONARY. The dictionary is a description of the type(s) of RECORDS to be stored in the database. A record is a collection of FIELDS. Each field contains a particular data value. For example, a record might contain 4 fields: name, social security number, rate of pay and number of tax exemptions. Within the dictionary each field is given a name, a data type code and a length. Also included for each field is a prompt string. The prompt string is a descriptive message which will be used by RMS when displaying the record as a form on the CRT screen. Optionally, the dictionary may include restrictions on the values which are to be allowed in a field. For example, a field which is to contain a grade from a class might be restricted to values of A,B,C,D or F. The type code of the field also determines the particular data values that can be stored. The RMS field types are A(lphanumeric), N(umeric), M(oney) and D(ate).

To sum up the function of the dictionary file: it tells what fields are to be stored in each record of the database, it tells what type of data each field may store, it tells what values are allowed in each field, and it determines what the record will look like when displayed as a form to be filled out on the CRT screen. RMS allows a database in which every record has the same format; or a database with two types of records. In this case there are primary records (which all have the same format), and each primary record may own any number of secondary records. The format of the secondary records is different from that of the primary records. An example of this organization is a database with a primary record for each customer, with a secondary record for each of the customer's invoices.

The dictionary may be entered with any text editor as a text file with the suffix ".DIC".

INITIALIZING AN RMS DATABASE

Once the dictionary file is entered on disk, the RMSNEW program is used to create an "empty" database file with the appropriate characteristics. The user need only supply the maximum length record, and the maximum number of records to be stored in the database. The database file itself will be in standard text file format with fixed length records. Thus it is easy to access the database directly from user written BASIC, PASCAL, etc. programs. RMS itself uses "hash coding" to locate individual records in the file. Hash coding is simply a formulatic method of scattering the data records all over the file in such a manner as to allow extremely rapid access to any particular record if its key is known. The hash coding formula is expiained and given as a BASIC subroutine in the RMS manual.

GETTING RECORDS IN AND OUT

The RMS record editor allows the operator to use the CRT screen as a window to the database. The screen will contain a form fill-out type of display that represents a record from the database. The user can add, change or delete records, look up information by record key, or browse through the file in random order or a specified order. RMS will insure that all data entered is valid for the type of field and for any restrictions placed on the field by the dictionary. This program serves the purpose of data entry, data validation, and online database access. RMS can look up any record in the database instantly and display it in the form on the screen. The form is made up of the prompt strings found in the data dictionary so it can look exactly as the implementor specifies. The cursor will come to rest only in places on the screen where the operator should type. It will not allow invalid data to be entered. The operator uses special function keys to move from field to field, enter data and cause records to be read or written to disk. As supplied RMS is configured for the SwTPC CT-82. However it is easy to re-configure it for most other types of CRT terminals.

GETTING PRINTED REPORTS

RMS includes a report writer facility. To use it you need only create a text file called a REPORT SPEC. The report spec is a few simple lines which tell what records from the file are to be printed, in what order they are to appear, and what the exact format of the printed output is to be. In general, the user will create many different report specs for the database. Each report spec will describe one of the printed reports to be produced on command. The following functions may be selected in any combination when creating a report spec: report title page; report wrap-up page; process records in any order; adjust to any page size; skip page perforations; include or exclude records as determined by field values; page headers; complete control over print format including the ability to print any number of lines per record; field totaling and sub-totaling; record counting; system date and page numbering; control breaks; and record group-end summaries.

SORTING AND INDEXING

RMS also includes an indexing utility which can be used to create an index on any field within the records. This index can be used to drive the RMS record editor or the report writer. This gives the user complete control over the order of record processing. The index utility is essentially an efficient sort/merge which produces an ordered list of record keys. The output is in text file format so indexes can be accessed or even created by user-written BASIC programs.

REORGANIZING THE DATABASE

The RMSCOPY utility is used to move the contents of one database file to another. It can perform one or more of the following functions in the process: merge two database files; post a transaction database to master database; expand or contract a file; or change the field and/or record layout of a file. This means that if, after the database is in use, new fields need to be added to the file, it can be doneby a simple dictionary change and RMSCOPY.

WHO CAN USE DBMS

Anyone who has a record-keeping function, be it accounting, customer lists, inventory, client control, student records or whatever, should consider the database management approach. It will normally be the most efficient route to a working system. And it will produce a system which can change to keep up with changing requirements. With RMS even the relative novice can implement a sophisticated database management system that is custom tailored to his specific needs. The experienced programmer can use RMS to leap far ahead in system implementation, skipping over many hours of unnecessary programming to do the common tasks that every system needs. His time can be spent directly on the programs that perform the unique processing of the particular system.

RMS is available for a 6809 Flex-based system from Washington Computer Services, 3028 Silvern Lane, Bellingham WA 98225. It is also available from Southwest Technical Products Corporation.

STUDENT Grade

'JW' stands for 'John Wood', the founder of Quincy, IL, and the namesake of one of the most innovative Community Colleges in America. 'JWCC' (as it is known) has teamed up with Harris Corp., one of the world's leading manufacturers of broadcast equipment, to provide a 2 year Associate Degree program in Broadcast Engineering Technology.

Students take support courses such as Math and English at local colleges, but the core courses are taught inside the Harris plant, where they have access to the same state of the art equipment (such as Spectrum Analyzers,Logic Analyzers,etc.,as is used on the 'front line'.

The Harris Instructional Staff never get behind

the state ot the art, as does happen in conventional Tech Schools, because they are continuously learning new products to teach to domestic and international students on a continuing basis.

This series of programs, written in TSC's XBASIC on a SWTPC 69A with 40 K of memory, was designed to take the drudgery out of keeping up with the grades for the John Wood students. With this series of programs a class roster can be created, and then added to or otherwise edited. After every editing job, the file is sorted in alphabetical order. The SCORE program grades a multiple-choice test. It call the roster for the class, asks you whether the student was present or absent, the questions missed, and for a personal code number known only to the student.(It is illegal to post grades by name or even by Social Security #, but grades can be posted by a personal code)

The SCORE program then analyzes each test item, and tells you what percentage of the class got it correctly. They are then categorized as 'simple','easy','moderate', 'hard',or 'difficult'. The printout lets you scan the results for suggested placement of items in the future. It has been shown that it is best to start out with easy questions and work up to hard ones.

The test grades are listed in three categories: 'raw', which is the actual grade, 'percentile', which is the 'normalizing' procedure to rank a student as to where he stands in relation to the mythical 'average' on ye olde bell-shaped-curve (the top of the curve=50), and the final classification is 'adjusted' which just shifts the bell curve upwards to where the top of the curve is at 75. (Students don't like to be called '50', but '75' they can live with.

The POST program tabulates any previuosly given test in any form you may want it for posting. Sorted by grade, by name, by SS#, by 'code' number, by percentile, etc.

The ISSUE program prints out a mailing list from the CLASS program, pulls the grades for each student, and gives a status report that can be mailed to each student.

The program is still in the works and is by no means complete but this is what I have so far. I keep an 'item' file on each test question I give, and a record of it's past performance.

I plan to put each item on a disk file, and each time a test is graded, the results will be added to the file. In the future I will be able to pick out which items I want and have the computer print me out a test. In fact, I will be able to specify a class average and have the computer pick my items that are most likely to give me that average. (If the class gets too cocky, for instance, and think they are hot-shots....I can always pull them down a peg or two.)

If you need any help with the administration of testing and record-keeping at your school via computer, I am only a phone-call away.
testing and record-keeping at your school via computer, I am only a phone-call away.

Martin J Petersen, Jr
327 Elm Street
Qunicy, Illinois 62301
(217)-224-6108

Senior Broadcast Technology Instructor
Harris Corporation
Quincy, Illinois 62301
(217)-222-8200

```
4000 REM "JM-GRADE.BAS" - MARTIN J. PETERSEN, JR. - AUGUST 10, 1980
4005 FOR A=1 TO 16:PRINT:NEXT A
4010 PRINT "THIS PROGRAM WILL GRADE A M.C. TEST AND SAVE RESULTS ON DISK-"
4015 PRINT "THE PROGRAM THEN TRANSFERS TO 'JM-PTEST', WHERE THE"
4020 PRINT "RESULTS OF THIS OR ANY OTHER TEST MAY BE REPORTED."
4025 PRINT
4030 GOSUB 4385
4035 OPEN OLD I$ AS 1
4040 INPUT#1,I$,C$,S$,H$,CN$,S
4045 DIM L$(S),F$(S),SS$(S),U(S)
4050 FOR A1=1 TO S
4055 INPUT #1,L$(A1),F$(A1),AA$,CT$,ST$,ZP,PH$,SS$(A1)
4060 NEXT A1
4065 CLOSE 1
4070 INPUT "TEST NUMBER (AS: 'FIRST', 'SECOND', ETC.)",T$
4075 INPUT "DATE (AS: 9/14/79 - NO COMMAS)",D$
4080 LET DT$="."
4085 LET T$=I$+DT$+CN$
4090 INPUT "# OF TEST ITEMS",Q
4095 DIM A$(S),X(Q,S),P(S),B(S),R(S),J(S),X1(Q+1,S+1)
4100 DIM TN1(S)
4105 FOR S1=1 TO S
4110 PRINT
4115 PRINT L$(S1);", ";F$(S1)
4120 INPUT "ABSENT (A) OR PRESENT (P)";A$(S1)
4125 IF A$(S1)="P" THEN INPUT "TEST #";TN1(S1)
4130 IF A$(S1)<>"P" THEN GOTO 4395
4135 IF A$(S1)="A" THEN GOTO 4170
4140 FOR Q1=1 TO Q
4145 INPUT "QUESTION # MISSED: (ENTER '0' AFTER LAST ENTRY)";I
4150 IF I=0 THEN 4170
4155 IF I>Q THEN PRINT "INVALID ENTRY":GOTO 4145
4160 LET X(I,S1)=1
4165 NEXT Q1
4170 NEXT S1
4175 PRINT:PRINT
4180 FOR S2=1 TO S
4185 FOR Q2=1 TO Q
4190 IF X(Q2,S2)=1 THEN X1(Q+1,S2)=X1(Q+1,S2)+1
4195 IF X(Q2,S2)=1 THEN X1(Q2,S+1)=X1(Q2,S+1)+1
4200 NEXT Q2
4205 LET U(S2)=Q-X1(Q+1,S2)
4210 LET R(S2)=U(S2)*100/Q
4215 IF A$(S2)="A" THEN R(S2)=0
4220 NEXT S2
4225 LET SP=S
4230 FOR S3=1 TO S
4235 IF A$(S3)="A" THEN SP=SP-1
4240 NEXT S3
4245 GOSUB 4405
4250 FOR S4=1 TO S
4255 IF A$(S4)="A" THEN 4295
4260 FOR S5=1 TO S
4265 IF A$(S5)="A" THEN 4280
4270 IF R(S5)<R(S4) THEN B(S4)=B(S4)+1
4275 IF R(S5)=R(S4) THEN E(S4)=E(S4)+1
4280 NEXT S5
4285 LET P(S4)=(B(S4)+E(S4)/2)/SP*100
4290 LET J(S4)=10*U(S4)+75
4295 NEXT S4
4300 PRINT "DATA IS NOW BEING RECORDED ON DISK"
4305 OPEN NEW T$ AS 2
4310 PRINT#2,I$;",";C$;",";S$;",";H$;",";T$;",";D$;",";Q;",";S
4315 PRINT#2,D$;",";T$;",";Q
4320 FOR S7=1 TO S
4325 PRINT#2,L$(S7);",";F$(S7);",";A$(S7);",";TN1(S7);",";[illegible]
4330 PRINT#2,R(S7);",";P(S7);",";J(S7);",";U(S7);",";SS$(S7)
4335 NEXT S7
4340 FOR S8=1 TO S
4345 FOR Q8=1 TO Q
4350 PRINT#2,X1(Q8,S8)
4355 NEXT Q8
4360 NEXT S8
4365 CLOSE 2
4370 PRINT
4375 PRINT "DATA IS NOW ON DISK UNDER FILENAME: ";T$
4380 CHAIN "JM-PTEST.BAS"
4385 INPUT "WHAT IS FILENAME OF CLASS (AS: DEC23080.PWR)";I$
4390 RETURN
4395 IF A$(S1)<>"A" THEN PRINT "INVALID ENTRY":GOTO 4120
4400 IF A$(S1)="A" THEN GOTO 4135
4405 FOR O=1 TO S
4410 IF A$(O)="A" THEN 4420
4415 LET U=U+U(O)
4420 NEXT O
4425 LET U=U/SP
4430 FOR H=1 TO S
4435 IF A$(H)="A" THEN 4455
4440 LET V=U(H)-U
4445 LET V=V*V
4450 LET W=W+V
4455 NEXT H
4460 LET D=SQR(W/SP)
4465 FOR I=1 TO S
4470 IF A$(I)="A" THEN 4485
4475 IF U(I)=U THEN U(I)=0:GOTO 4485
4480 LET U(I)=(U(I)-U)/D
4485 NEXT I
4490 RETURN
```

```
10 REM "JM-MENU.BAS" - MARTIN J. PETERSEN, JR. - AUGUST 10, 1980
20 FOR A=1 TO 16:PRINT:NEXT A
30 PRINT "THIS IS THE 'JOHN WOOD' PROGRAM FOR ADMINISTERING TESTS AND GRADES"
40 PRINT
50 PRINT TAB(10);"1. . .CREATE A NEW CLASS ROSTER"
60 PRINT TAB(10);"2. . .EDIT EXISTING CLASS ROSTER"
70 PRINT TAB(10);"3. . .PRINT AN EXISTING CLASS ROSTER"
80 PRINT TAB(10);"4. . .GRADE A NEW TEST"
90 PRINT TAB(10);"5. . .PRINT A TEST GRADE REPORT"
100 PRINT TAB(10);"6. . .PRINT A CLASS GRADE REPORT"
110 PRINT TAB(10);"7. . .PRINT A STUDENT GRADE REPORT"
120 PRINT TAB(10);"8. . .PRINT MAILING LABELS"
130 PRINT TAB(10);"9. . .CREATE TEST ITEM FILES"
140 PRINT TAB(9);"10. . .EDIT TEST ITEM FILES"
150 PRINT TAB(9);"11. . .PRINT A TEST FROM ITEM FILES"
160 PRINT TAB(9);"12. . .EXIT PROGRAM"
170 PRINT
180 INPUT "WHAT IS YOUR SELECTION (MENU #) ";N
190 ON N GOTO 200,210,220,230,240,250,260,270,280,290,300,310
200 CHAIN "JM-NROST.BAS"
210 CHAIN "JM-REDIT.BAS"
220 CHAIN "JM-PRINT.BAS"
230 CHAIN "JM-GRADE.BAS"
240 CHAIN "JM-PTEST.BAS"
250 CHAIN "JM-PCLAS.BAS"
260 CHAIN "JM-PSTUD.BAS"
270 CHAIN "JM-MAIL.BAS"
280 CHAIN "JM-OMAKE.BAS"
290 CHAIN "JM-OCHNG.BAS"
300 CHAIN "JM-OPRIN.BAS"
310 FOR B=1 TO 16:PRINT:NEXT B
320 PRINT "IT HAS BEEN MY PLEASURE TO SERVE YOU. . ."
330 PRINT "PLEASE CALL ON ME AGAIN WHEN YOU NEED ME."
340 PRINT
350 PRINT "HAVE A NICE DAY. . ."
360 END
```

```
2000 REM "JM-REDIT.BAS" - MARTIN J. PETERSEN, JR. - AUGUST 10, 1980
2005 FOR A=1 TO 16:PRINT:NEXT A
2010 PRINT "THIS PROGRAM WILL EDIT EXISTING CLASS ROSTERS"
2015 PRINT
2020 INPUT "WHAT IS FILENAME OF ROSTER YOU WISH TO EDIT";I$
2025 OPEN OLD I$ AS 1
2030 INPUT#1,I$,C$,S$,H$,CN$,S
2035 DIM L$(S),F$(S),AA$(S),CT$(S),ST$(S),ZP(S),PH$(S),SS$(S)
2040 DIM W$(S)
2045 FOR B=1 TO S
2050 INPUT #1,L$(B),F$(B),AA$(B),CT$(B),ST$(B),ZP(B),PH$(B),SS$(B)
2055 NEXT B
2060 CLOSE 1
2065 INPUT "DO YOU WISH TO SCAN THE FILE";SC$
2070 IF ASC(SC$)=ASC("Y") THEN GOSUB 2650
2075 INPUT "CHOOSE 'MODIFY (M)', 'DELETE (D)', 'ADD (A)', OR 'EXIT (E)'";E$
2080 IF E$="M" THEN 2100
2085 IF E$="D" THEN 3215
2090 IF E$="A" THEN 2770
```

```
2095 IF B$<>"Y" THEN 2700, ELSE 2065
2100 IF ASC(M$)=ASC("Y") THEN 2215
2105 INPUT "DO YOU WISH TO MODIFY HEADER",M$
2110 IF ASC(M$)<>ASC("Y") THEN 2215
2115 PRINT
2120 PRINT Z$;TAB(30);"(Z$)"
2125 PRINT C$;TAB(30);"(C$)"
2130 PRINT S$;TAB(30);"(S$)"
2135 PRINT H$;TAB(30);"(H$)"
2140 PRINT CN$;TAB(30);"(CN$)"
2145 PRINT
2150 INPUT "COURSE #(AS: EET 100)",C$
2155 INPUT "SEMESTER(AS: WINTER 1980)",S$
2160 INPUT "DAY OR NIGHT",H$
2165 INPUT "3-LETTER CODE-NAME",CN$
2170 LET Z$=LEFT$(H$,1)
2175 LET Z$=Z$+LEFT$(C$,1)
2180 LET Z$=Z$+MID$(C$,5,3)
2185 LET Z$=Z$+LEFT$(S$,1)
2190 LET Z$=Z$+RIGHT$(S$,2)
2195 LET DT$="."
2200 LET Z$=Z$+DT$+CN$
2205 PRINT
2210 GOTO 2065
2215 INPUT "WHAT IS STUDENT'S LAST NAME",L$
2220 PRINT
2225 FOR C=1 TO S
2230 IF L$(C)=L$ THEN GOTO 2475
2235 NEXT C
2240 FOR D=1 TO (S-1)
2245 SWAP L$(D),L$(D+1)
2250 SWAP F$(D),F$(D+1)
2255 SWAP AA$(D),AA$(D+1)
2260 SWAP CT$(D),CT$(D+1)
2265 SWAP ST$(D),ST$(D+1)
2270 SWAP ZP(D),ZP(D+1)
2275 SWAP PH$(D),PH$(D+1)
2280 SWAP SS$(D),SS$(D+1)
2285 NEXT D
2290 LET S=S-1:PRINT "STUDENT RECORD DELETED": GOTO 2065
2295 DIM XL$(S+1),XF$(S+1),XA$(S+1),XC$(S+1),XS$(S+1)
2300 DIM XZ(S+1),XP$(S+1),XN$(S+1)
2305 PRINT
2310 INPUT "NEW STUDENT'S LAST NAME",XL$(S+1)
2315 INPUT "FIRST NAME",XF$(S+1)
2320 INPUT "ADDRESS",XA$(S+1)
2325 INPUT "CITY",XC$(S+1)
2330 INPUT "STATE",XS$(S+1)
2335 INPUT "ZIP CODE",XZ(S+1)
2340 INPUT "PHONE",XP$(S+1)
2345 INPUT "SOCIAL SECURITY #",XN$(S+1)
2350 LET S=S+1
2355 FOR D=1 TO (S-1)
2360 LET XL$(D)=L$(D)
2365 LET XF$(D)=F$(D)
2370 LET XA$(D)=AA$(D)
2375 LET XC$(D)=CT$(D)
2380 LET XS$(D)=ST$(D)
2385 LET XZ(D)=ZP(D)
2390 LET XP$(D)=PH$(D)
2395 LET XN$(D)=SS$(D)
2400 NEXT D
2405 FOR E=1 TO (S-1)
2410 FOR F=1 TO (S-E)
2415 IF XL$(F)<XL$(F+1) THEN 2460
2420 SWAP XL$(F),XL$(F+1)
2425 SWAP XF$(F),XF$(F+1)
2430 SWAP XA$(F),XA$(F+1)
2435 SWAP XC$(F),XC$(F+1)
2440 SWAP XS$(F),XS$(F+1)
2445 SWAP XZ(F),XZ(F+1)
2450 SWAP XP$(F),XP$(F+1)
2455 SWAP XN$(F),XN$(F+1)
2460 NEXT F
2465 NEXT E
2470 GOTO 2595
2475 IF L$(C)<>L$ THEN GOSUB 2700
2480 IF B$="D" THEN 2290
2485 PRINT:PRINT L$(C);", ";F$(C);TAB(40);"(L$,F$)"
2490 PRINT AA$(C);TAB(40);"(AA$)"
2495 PRINT CT$(C);", ";ST$(C);TAB(40);"(CT$,ST$)"
2500 PRINT ZP(C);TAB(40);"(ZP)"
2505 PRINT
2510 PRINT PH$(C);TAB(40);"(PH$)"
2515 PRINT SS$(C);TAB(40);"(SS$)"
2520 PRINT

2525 INPUT "ENTER MOD. CODE: (L$, F$, AA$, CT$, ST$, ZP, PH$, SS$)",M$
2530 INPUT "ENTER NEW INFORMATION",I$
2535 IF M$="L$" THEN L$(C)=I$
2540 IF M$="F$" THEN F$(C)=I$
2545 IF M$="AA$" THEN AA$(C)=I$
2550 IF M$="CT$" THEN CT$(C)=I$
2555 IF M$="ST$" THEN ST$(C)=I$
2560 IF M$="ZP" THEN ZP(C)=I$
2565 IF M$="PH$" THEN PH$(C)=I$
2570 IF M$="SS$" THEN SS$(C)=I$
2575 PRINT
2580 INPUT "IS THERE ANOTHER MODIFICATION OF THIS SAME RECORD",Q$
2585 IF ASC(Q$)=ASC("Y") THEN 2525
2590 GOTO 2065
2595 OPEN NEW Z$ AS 3
2600 PRINT#3,Z$;",";C$;",";S$;",";H$;",";CN$;",";S
2605 FOR A=1 TO S
2610 PRINT #3,XL$(A);",";XF$(A);",";XA$(A);",";XC$(A);",";
2615 PRINT #3,XS$(A);",";XZ(A);",";XP$(A);",";XN$(A)
2620 NEXT A
2625 CLOSE 3
2630 INPUT "ARE THERE OTHER FILE MODIFICATIONS",Q$
2635 IF ASC(Q$)=ASC("Y") THEN CHAIN "JA-EDIT.BAS"
2640 PRINT:PRINT "DATA IS ON DISK UNDER FILENAME ";Z$
2645 CHAIN "JA-MENU.BAS"
2650 FOR T=1 TO S
2655 PRINT:PRINT L$(T);", ";F$(T)
2660 PRINT AA$(T)
2665 PRINT CT$(T);", ";ST$(T)
2670 PRINT ZP(T)
2675 PRINT "PHONE: ";PH$(T)
2680 PRINT "SOCIAL SECURITY #";SS$(T)
2685 PRINT
2690 NEXT T
2695 RETURN
2700 FOR C=1 TO S-1
2705 IF L$(C)=L$(C+1) THEN B=B+1
2710 NEXT C
2715 PRINT "THERE ARE ";B+1;" ";L$;"'S ON FILE."
2720 INPUT "WHAT IS STUDENT'S FIRST NAME [use 'END' to cancel]",F$
2725 FOR F=C TO C+B
2730 IF F$(F)=F$ THEN C=F:RETURN
2735 NEXT F
2740 PRINT "COULD NOT FIND ";F$
2745 PRINT "[use 'END' to cancel]"
2750 LET B=0:GOTO 2065
2755 OPEN NEW Z$ AS 2
2760 PRINT#2,Z$;",";C$;",";S$;",";H$;",";CN$;",";S
2765 FOR A=1 TO S
2770 PRINT #2,L$(A);",";F$(A);",";AA$(A);",";CT$(A);",";
2775 PRINT #2,ST$(A);",";ZP(A);",";PH$(A);",";SS$(A)
2780 NEXT A
2785 CLOSE 2
2790 PRINT
2795 PRINT "DATA IS ON DISK UNDER FILENAME ";Z$
2800 CHAIN "JA-MENU.BAS"


3000 REM 'JA-ROSTER.BAS' - MARTIN J. PETERSEN, JR. - AUGUST 18, 1980
3005 FOR A=1 TO 8 :PRINT:NEXT A
3010 PRINT "THIS PROGRAM PRINTS A CLASS LIST (ROSTER) FOR ANY"
3015 PRINT "CLASS ON DISKFILE."
3020 INPUT "FILENAME (AS: DE123S80.PWR)",Z$
3025 OPEN OLD Z$ AS 1
3030 INPUT#1,Z$,C$,S$,H$,CN$,S
3035 DIM L$(S),F$(S),AA$(S),CT$(S),ST$(S),ZP(S),PH$(S),SS$(S)
3040 FOR B=1 TO S
3045 INPUT#1,L$(B),F$(B),AA$(B),CT$(B),ST$(B),ZP(B),PH$(B),SS$(B)
3050 NEXT B
3055 CLOSE 1
3060 INPUT "TERMINAL (T), PRINTER (P), OR TTY (Y)",Y$
3065 IF ASC(Y$)=ASC("P") THEN OPEN "PRINTER.CMD" AS 0
3070 IF ASC(Y$)=ASC("Y") THEN OPEN "TTY.CMD" AS 0
3075 PRINT#0,TAB(30);"CLASS ROSTER"
3080 PRINT#0
3085 PRINT#0,Z$;", ";C$;", ";S$;", ";H$;" CLASS, ";S;" STUDENTS"
3090 PRINT#0,"CLASS CODE-NAME IS ";CN$
3095 PRINT#0
3100 LET X=1:Y=6
3105 LET Z=Z+1
3110 FOR C=X TO Y
3115 IF C>S THEN 3330
3120 PRINT#0,C;". ";
3125 IF (C+6)>S THEN PRINT#0:GOTO 3145
3130 PRINT#0,TAB(24);(C+6);". ";
3135 IF (C+12)>S THEN PRINT#0:GOTO 3145
3140 PRINT#0,TAB(40);(C+12);". "
3145 IF C>S THEN 3330
3150 PRINT#0,L$(C);", ";F$(C);
3155 IF (C+6)>S THEN PRINT#0:GOTO 3175
3160 PRINT#0,TAB(24);L$(C+6);", ";F$(C+6);
3165 IF (C+12)>S THEN PRINT#0:GOTO 3175
3170 PRINT#0,TAB(40);L$(C+12);", ";F$(C+12)
3175 IF C>S THEN 3330
3180 PRINT#0,AA$(C);
3185 IF (C+6)>S THEN PRINT#0:GOTO 3205
3190 PRINT#0,TAB(24);AA$(C+6);
3195 IF (C+12)>S THEN PRINT#0:GOTO 3205
3200 PRINT#0,TAB(40);AA$(C+12)
3205 IF C>S THEN 3330
3210 PRINT#0,CT$(C);", ";ST$(C);
3215 IF (C+6)>S THEN PRINT#0:GOTO 3235
3220 PRINT#0,TAB(24);CT$(C+6);", ";ST$(C+6);
3225 IF (C+12)>S THEN PRINT#0:GOTO 3235
3230 PRINT#0,TAB(40);CT$(C+12);", ";ST$(C+12)
3235 IF C>S THEN 3330
3240 PRINT#0,TAB(12);ZP(C);
3245 IF (C+6)>S THEN PRINT#0:GOTO 3265
3250 PRINT#0,TAB(35);ZP(C+6);
3255 IF (C+12)>S THEN PRINT#0:GOTO 3265
3260 PRINT#0,TAB(51);ZP(C+12)
3265 IF C>S THEN 3330
3270 PRINT#0,"PHONE: ";PH$(C);
3275 IF (C+6)>S THEN PRINT#0:GOTO 3295
3280 PRINT#0,TAB(24);"PHONE: ";PH$(C+6);
3285 IF (C+12)>S THEN PRINT#0:GOTO 3295
3290 PRINT#0,TAB(40);"PHONE: ";PH$(C+12)
3295 IF C>S THEN 3330
3300 PRINT#0,"S.S.# ";SS$(C);
3305 IF (C+6)>S THEN PRINT#0:GOTO 3325
3310 PRINT#0,TAB(24);"S.S.# ";SS$(C+6);
3315 IF (C+12)>S THEN PRINT#0:GOTO 3325
3320 PRINT#0,TAB(40);"S.S.# ";SS$(C+12)
3325 PRINT#0
3330 PRINT#0
3335 NEXT C
3340 IF S<(18*Z)+1 THEN 3390
3345 FOR D=1 TO 8
3350 PRINT#0
3355 NEXT D
3360 PRINT#0,TAB(20);"CLASS ROSTER (CONTINUED)"
3365 PRINT#0
3370 PRINT#0,Z$;", ";C$;", ";S$;", ";H$;" CLASS, ";S;" STUDENTS"
3375 PRINT#0
3380 LET X=(18*Z)+1:Y=X+6
3385 GOTO 3105
3390 PRINT "LIST COMPLETED"
3395 CLOSE 0
3400 CHAIN "JA-MENU.BAS"

1000 REM 'JA-ENROLL.BAS' - MARTIN J. PETERSEN, JR. - AUGUST 18, 1980
1005 FOR A=1 TO 16:PRINT:NEXT A
1010 PRINT "THIS PROGRAM CREATES A ROSTER FOR A NEW CLASS"
1015 PRINT
1020 GOSUB 1160
1025 PRINT
1030 PRINT "THIS CLASS WILL BE KNOWN BY THE DISK FILENAME OF ";Z$
1035 PRINT
1040 DIM L$(35),F$(35),AA$(35),CT$(35),ST$(35),ZP(35),PH$(35),SS$(35)
1045 LET S=S+1
1050 PRINT "(TYPE 'F' WHEN FINISHED)"
1055 INPUT "STUDENT'S LAST NAME",L$(S)
1060 IF L$(S)="F" THEN S=S-1:GOTO 1110
1065 INPUT "FIRST NAME",F$(S)
1070 INPUT "ADDRESS",AA$(S)
1075 INPUT "CITY",CT$(S)
1080 INPUT "STATE",ST$(S)
1085 INPUT "ZIPCODE",ZP(S)
1090 INPUT "PHONE NUMBER",PH$(S)
1095 INPUT "SOCIAL SECURITY #",SS$(S)
1100 PRINT
1105 GOTO 1045
1110 OPEN NEW Z$ AS 1
1115 PRINT#1,Z$;",";C$;",";S$;",";H$;",";CN$;",";S
1120 FOR A=1 TO S
1125 PRINT #1,L$(A);",";F$(A);",";AA$(A);",";CT$(A);",";
1130 PRINT #1,ST$(A);",";ZP(A);",";PH$(A);",";SS$(A)
1135 NEXT A
1140 CLOSE 1
1145 PRINT
1150 PRINT "DATA IS ON DISK UNDER FILENAME ";Z$
1155 CHAIN "JA-MENU.BAS"
1160 INPUT "COURSE #(AS: EET 100)",C$
1165 INPUT "SEMESTER(AS: WINTER 1980)",S$
1170 INPUT "DAY OR NIGHT",H$
1175 LET Z$=LEFT$(H$,1)
1180 LET Z$=Z$+LEFT$(C$,1)
1185 LET Z$=Z$+MID$(C$,5,3)
1190 LET Z$=Z$+LEFT$(S$,1)
1195 LET Z$=Z$+RIGHT$(S$,2)
1200 INPUT "WHAT 3-LETTER 'CODE-NAME' WILL BE GIVEN THIS CLASS",CN$
1205 LET DT$="."
1210 LET Z$=Z$+DT$+CN$
1215 RETURN

5000 REM 'JA-FIRST.BAS' - MARTIN J. PETERSEN, JR. - AUGUST 20, 1980
5005 FOR A=1 TO 16:PRINT:NEXT A
5010 PRINT "THIS PROGRAM WILL PRINT THE RESULTS OF ANY PREVIOUS TEST"
5015 PRINT "ARRANGED IN ORDER BY NAME, TEST #, OR SCORE"
5020 PRINT
5025 INPUT "WHAT IS FILENAME OF TEST (AS: "FIRST.PWR", E.C.)",T$
5030 OPEN OLD T$ AS 1
5035 INPUT#1,Z$,C$,S$,H$,T$,D$,E$,Q,S,P
5040 DIM L$(S),F$(S),A(S),T(S),B(S),R(S),U(S),X(Q+1,P+1),W(S)
5045 DIM U(S)
5050 FOR A=1 TO S
5055 INPUT#1,L$(A),F$(A),A(A),T(A),R(A),P(A),J(A),U(A),W(A)
5060 NEXT A
5065 FOR A2=1 TO S
5070 FOR A3=1 TO Q
5075 INPUT #1,X(A3,A2)
5080 NEXT A3
5085 NEXT A2
5090 FOR S2=1 TO S
5095 FOR Q2=1 TO Q
5100 IF X(Q2,S2)=1 THEN X(Q+1,S2)=X(Q+1,S2)+1
5105 IF X(Q2,S2)=1 THEN X(Q2,S+1)=X(Q2,S+1)+1
5110 NEXT Q2
5115 NEXT S2
5120 GOSUB 5725
5125 PRINT
5130 PRINT "CHOOSE SORTING METHOD BY NUMBER:"
5135 PRINT
5140 PRINT "LAST NAME . . . 1"
5145 PRINT "TEST # . . . . . 2"
5150 PRINT "RAW SCORE . . . 3"
5155 PRINT "TITLE . . . . . 4"
5160 PRINT "ADJUSTED SCORE . 5"
5165 PRINT "S.S. # . . . . . 6"
5170 PRINT
5175 INPUT "WHAT IS DESIRED SORTING METHOD",X
5180 GOSUB 5440
5185 PRINT "THEN LIST ABOVE, CHOOSE 'NAME', 'TEST #', OR 'S.S. #'"
5190 INPUT "WHAT IS DESIRED POSTING METHOD",Y
5195 ON Y GOTO 5565,5570,5575,5575,5575,5580
5200 PRINT
5205 INPUT "TERMINAL (T), PRINTER (P), OR TELETYPE (Y)",Y$
5210 IF Y$="P" THEN OPEN "PRINTER.CMD" AS 0
5215 IF Y$="Y" THEN OPEN "TTY.CMD" AS 0
5220 PRINT#0
5225 PRINT#0,Z$;", ";C$;", ";S$;", ";H$;" EXAM, ";T$
5230 PRINT#0,"SORTED BY ";
5235 ON X GOTO 5625,5630,5640,5650,5665,5670
5240 PRINT#0,TAB(30);"POSTED BY ";
5245 ON Y GOTO 5685,5690,5695,5695,5695,5700
5250 PRINT#0
```

```
5255 PRINT#0,[illegible]
5260 PRINT [illegible]
5265 FOR S6=1 TO 5
5270 ON [illegible] GOTO 5585,5595,5605,5605,5605,5615
5275 IF [illegible] THEN [illegible]:GOTO 5315
5280 PRINT#0,USING [illegible]
5285 GOSUB 5740
5290 PRINT#0,USING [illegible]
5295 GOSUB 5705
5300 PRINT#0,USING [illegible]
5305 GOSUB 5775
5310 LET [illegible]
5315 NEXT S6
5320 PRINT#0
5325 PRINT#0,[illegible]
5330 PRINT#0,USING [illegible]
5335 LET [illegible]
5340 PRINT#0
5345 PRINT#0,"ITEM DIFFICULTY: % OF CLASS RESPONDING CORRECTLY"
5350 PRINT#0
5355 FOR I=1 TO 0
5360 LET [illegible]
5365 PRINT#0,[illegible]
5370 [illegible]
5375 LET [illegible]
5380 IF [illegible] THEN PRINT#0,TAB(4);"DIFFICULT":GOTO 5405
5385 IF [illegible] THEN PRINT#0,TAB(36);"HARD":GOTO 5405
5390 IF [illegible] THEN PRINT#0,TAB(26);"MODERATE":GOTO 5405
5395 IF [illegible] THEN PRINT#0,TAB(20);"EASY":GOTO 5405
5400 PRINT#0,TAB(12);"SIMPLE"
5405 NEXT I
5410 PRINT#0
5415 PRINT#0,"AVERAGE ITEM DIFFICULTY:"[illegible]
5420 [illegible]
5425 LET [illegible]
5430 CLOSE 0
5435 INPUT [illegible]
5440 IF [illegible] THEN 5125
5445 INPUT [illegible]
5450 IF [illegible] THEN 5025
5455 CHAIN "A:MENU,BAS"
5460 FOR E=1 TO [illegible]
5465 FOR F=1 TO [illegible]
5470 ON [illegible] GOTO 5535,5540,5545,5550,5555,5560
5475 SWAP L$(F),L$(F+1)
5480 SWAP F$(F),F$(F+1)
5485 SWAP A$(F),A$(F+1)
5490 SWAP TN(F),TN(F+1)
5495 SWAP R(F),R(F+1)
5500 SWAP P(F),P(F+1)
5505 SWAP J(F),J(F+1)
5510 SWAP SS$(F),SS$(F+1)
5515 SWAP U(F),U(F+1)
5520 NEXT F
5525 NEXT E
5530 RETURN
5535 IF L$(F)<L$(F+1) THEN 5520 ELSE 5475
5540 IF TN(F)<TN(F+1) THEN 5520 ELSE 5475
5545 IF R(F)>R(F+1) THEN 5520 ELSE 5475
5550 IF P(F)>P(F+1) THEN 5520 ELSE 5475
5555 IF J(F)>J(F+1) THEN 5520 ELSE 5475
5560 IF SS$(F)<SS$(F+1) THEN 5520 ELSE 5475
5565 LET W$="STUDENT":GOTO 5200
5570 LET W$="TEST #":GOTO 5200
5575 PRINT "[illegible] PLEASE READ THE QUESTION AGAIN":GOTO 5100
5580 LET W$="S.S. #":GOTO 5200
5585 PRINT#0,L$(S6);
5590 GOTO 5275
5595 IF [illegible]="A" THEN PRINT#0,[illegible]
5600 IF [illegible]="A" THEN 5315 ELSE 5275
5605 PRINT#0,"SCORE TALLY";
5610 GOTO 5275
5615 IF [illegible]="A" THEN PRINT#0,[illegible]
5620 IF [illegible]="A" THEN 5315 ELSE 5275
5625 PRINT#0,"STUDENT";
5630 GOTO 5240
5635 PRINT#0,"TEST #";
5640 GOTO 5240
5645 PRINT#0,"RAW SCORE";
5650 GOTO 5240
5655 PRINT#0,"PERCENTILE";
5660 GOTO 5240
5665 PRINT#0,"ADJUSTED SCORE";
5670 GOTO 5240
5675 PRINT#0,"S.S.#";
5680 GOTO 5240
5685 PRINT#0,"STUDENT":GOTO 5250
5690 PRINT#0,"TEST #":GOTO 5250
5695 PRINT#0,"SCORE":GOTO 5250
5700 PRINT#0,"S.S. #":GOTO 5250
5705 LET GR$="F"
5710 IF U(S6)>-2 THEN GR$="D"
5715 IF U(S6)>-1 THEN GR$="C"
5720 IF U(S6)>1 THEN GR$="B"
5725 IF U(S6)>2 THEN GR$="A"
5730 PRINT#0,TAB(35);GR$;
5735 RETURN
5740 LET GR$="F"
5745 IF R(S6)>59 THEN GR$="D"
5750 IF R(S6)>69 THEN GR$="C"
5755 IF R(S6)>79 THEN GR$="B"
5760 IF R(S6)>89 THEN GR$="A"
5765 PRINT#0,TAB(20);GR$;
5770 RETURN
5775 LET GR$="F"
5780 IF J(S6)>59 THEN GR$="D"
5785 IF J(S6)>69 THEN GR$="C"
5790 IF J(S6)>79 THEN GR$="B"
5795 IF J(S6)>89 THEN GR$="A"
5800 PRINT#0,TAB(50);GR$;
5805 PRINT#0,USING [illegible]
5810 RETURN
```

```
[illegible] 123 SPRING 1980 [illegible] 13
CARNES,Brian,[illegible],IL. 62347 ,645-3332,343-40-5847
CREGO,Alan,31 Country Club Drive,Quincy,IL. 62301 ,224-4495,157-52-5019
JEFFRIES,James,RR #1,Davis,IL. 62314 ,933-2003,346-60-6537
KAMP,Edwin,RR #1,Mozier,IL. 62070 ,232-1285,344-60-9922
MALANEY,Rower K.,209 Hereford Street,Payson,IL. 62360 ,240-3024,310-42-5850
MARTENS,Brad,2519 College,Quincy,IL. 62301 ,223-5575,352-50-6465
MENDENHALL,Mike,1231 Daniel Court,Quincy,IL. 62301 ,228-0822,211-30-7000
O'CONNELL,David,115 W.North,Mt. Sterling,IL. 62353 ,773-2060,352-26-1426
PHILLIPS,Mark,826 Adams,Quincy,IL. 62301 ,224-4825,331-46-8318
SCHULTE,Roch,1325 Locust,Quincy,IL. 62301 ,223-3774,340-50-7632
STEFFA,Robert,707 Kentucky - Apt 2,Quincy,IL. 62301 ,224-9450,347-50-4701
WARREN,Mark,[illegible] Hampshire,Quincy,IL. 62301 ,222-[illegible]
[illegible] IL. 62[illegible] ,[illegible]-4149

[illegible] 123 SPRING 1980 [illegible] 20 , 13 , 11
CARNES,Brian,P, 2 , 40 , 12.[illegible] , 48.0153[illegible] ,-.[illegible] ,343-40-5847
CREGO,Alan,P, 5 , 70 , 22.7272727272 , 71.1170[illegible] ,-.3002901372723 ,157-52-5019
JEFFRIES,James,P, 17 , 100 , [illegible] , [illegible] , 1.0022[illegible] ,346-60-6537
KAMP,Edwin,P, 22 , 90 , 50 , 77.2100007049[illegible] , .22100007049[illegible] ,344-60-9922
MALANEY,Rower K.,P, 1 , 35 , 4.545454545[illegible] , 49.7611410700 ,-3.5330050[illegible] ,310-42-5850
MARTENS,Brad,A, 0 , 0 , 0 , 0 , 20 ,352-50-6465
MENDENHALL,Mike,P, [illegible] , 75 , [illegible] , [illegible] ,-.[illegible] ,211-30-7000
O'CONNELL,David,P, 21 , 80 , 50 , 77.21[illegible] , .221[illegible] ,352-26-1426
PHILLIPS,Mark,P, 7 , 65 , 72.7272727272 , 80.2696510[illegible] , .5269651[illegible] ,331-46-8318
SCHULTE,Roch,P, 10 , 80 , 50 , 77.21[illegible] , .221[illegible] ,340-50-7632
STEFFA,Robert,A, 0 , 0 , 0 , 0 , 20 ,347-50-4701
WARREN,Mark,P, 2 , 70 , [illegible] , [illegible] , .0320502943692 ,[illegible]
[illegible],P, 4 , 85 , 72.7272727272 , [illegible] , .5269651064330 ,[illegible]-4149

0
0
[illegible] 123 SPRING 1980,[illegible],FINAL,[illegible],FINAL, 39 , 13 , 12
CARNES,Brian,P, 7 , 60 , 62.5 , 72.[illegible] ,-.[illegible]10970 ,[illegible]
CREGO,Alan,P, 9 , 64 , 29.1666666667 , 60.66314521307 ,-.6336085478693 ,157-52-5019
JEFFRIES,James,P, 3 , 64 , 29.1666666667 , 60.66314521307 ,-.[illegible] ,[illegible]
KAMP,Edwin,P, 9 , 90 , 97.5 , 92.6764096600 , 1.7676409660 ,344-60-9922
MALANEY,Rower K.,P, 10 , 64 , 29.1666666667 , 60.66314521307 ,-.[illegible] ,[illegible]
MARTENS,Brad,P, 1 , 80 , 79.1666666667 , 84.67200111609 , .9672004151609 ,[illegible]
MENDENHALL,Mike,A, 0 , 0 , 0 , 0 , 25 ,211-30-7000
O'CONNELL,David,P, 8 , 64 , 29.1666666667 , 60.66314521307 ,-.6336085478693 ,[illegible]-1426
PHILLIPS,Mark,P, 6 , 60 , 6.1666666667 , [illegible] ,-[illegible] ,[illegible]
SCHULTE,Roch,P, 5 , 60 , 62.5 , 72.66536928903 ,-.2334630710974 ,340-50-7632
STEFFA,Robert,P, 12 , 60 , 62.5 , 72.66536928903 ,-.2334630710974 ,347-50-4701
WARREN,Mark,P, 11 , 92 , 95.8333333333 , 96.67871374476 , 1.6787137447 ,[illegible]
[illegible],P, 3 , 64 , 29.1666666667 , 60.66314521307 ,-.6336085478693 ,355-60-4149
0
1
```

BIT Bucket

Peter F. Hills
Box 196
Somerset, Ca 95684
916-626-0749

September 2, 1980

Mr. Don Williams
68 Micro Journal
3018 Hamill Rd
PO Box 849
Hixson, TN 37343

Dear Mr. Williams:

I have been a subscriber of yours since the beginning of this year, and would like to commend you for the fine job you have done. I have gotten a lot of useful information out of '68' Micro Journal. Ron Anderson's FLEX USER NOTES have been particularly helpful, since I am a newcomer to FLEX.

I have put together an S-100 based system consisting of:

- Microdaya 6900 CPU board
- 48K RAM
- SD Systems Versafloppy I
- SSM PB1 Prom Programmer
- Integrand Mainframe
- 2 Memorex 652 Drives, modified to look like Shugarts
- Hazeltine 1500 Terminal
- Qume S3/45 Printer
- Misc custom interfaces, including a 19.2 Kbaud serial link to another homebrew computer using PACE.
- FLEX V 2.0 GENERAL for 6809, customized to the hardware.

For your information, I do not recommend the use of the Microdasys board. It is not S100 compatible without extensive modifications, and is in general not well thought out. The SSM Prom programmer is O.K. for 8080/Z80 but not for 6809, as it uses wait states for timing. The Versafloppy is excellent and well documented. FLEX is terrific! The documentation for the general version is very complete, and makes customization very straightforward.

I have since received a 6809 CPU board from Ackerman Digital Systems which looks good, but have not tried it out yet.

I have followed with some amusement the chauvinistic discourses in your publication regarding the advantages of the S-50 bus over the S-100. Let me just say that I made my decision to go with S-100 for one simple reason: it would allow me to run both 6809 and Z80 in the same mainframe, using off the shelf hardware. This is important to me in my work, as my clients often dictate which processor must be used in their systems. It turns out that I was pretty naive regarding that "off the shelf" part, but I still think I made the right decision. In fact, the only thing I question is whether I should have even bothered with 6809 for a while. Consider: Motorola still doesn't have a decent User's Manual. The price of 6809 chips is still high. Software is still scant. I have ordered the STYLUS word processor from Sonex Systems, but I'm not sure if it will support proportional printing, bidirectional printing or if it will in fact interface with my printer. I have also installed the 6809 FORTH version distributed by the Forth Interest Group, but haven't learned how to use it yet.

I do see some hope for the future, as usage of the 6809 increases. The processor itself has obvious advantages.

I hope I haven't offended anyone by my comments; I just wanted to deliver an honest opinion.

Sincerely

Peter F. Hills

Peter F. Hills

+++

Editor's Note:

Pete: Thanks first for the nice compliments concerning 68 Micro Journal, never get tired of receiving them.

We have also heard grumblings about the MicroDaSys 6809 CPU board. It is unfortunate but there has been practically no support for that type of configuration. Most all 68XX support is geared to the S50 bus. I personally know many who run 68XX devices on backplanes other than the S50; however, most spend more time getting everything 'together' rather then operating. For many this is the 'fun' of the thing, for others it is a pile of worms. I guess that our 'chauvinistic' leanings are due to the our desire to chug along with as little hassle as possible, ie.,S50 bus. Of course the Z80 has been mated to the S50 bus (we have one) but then it is a can of worms for us...no support!

I know also many 'consultants' who sell their services using both the 68XX and 80XX devices. Seems that the best route is a seperate backplane for each. However; I know of NO application using a 80XX device that is more efficient or cost effective than the same using a 68XX.

Re: Motorola 6809 manual, on this count you rate 100..the manual stinks (compared to past publications). The price of the 6809 seems reasonable and on the software you are off. There is more useful applications software for the 6809 than any other chip in it's class. The variety is not as great but the 'quality' is there...note products advertised in these pages. I do not know of any 80XX software that will begin to compare to Microware's OS-9™ or TSC's Uniflex™. And these are only a beginning!!

Again Pete I appreciate your thoughts, not even all of us who are exclusive 68XX, S50 agree among ourselves. But the one thing that has keep us a viable and GROWING group has not been the 68XX as much as it has been the S50 bus. After all who enjoys their ice cream with mustard and horseradish?

DMW

FULL SCREEN DISPLAYS
DR. E. M. PASS

Few programming systems for micro computers currently provide more than primitive support for full screen (two dimensional) display and input. This is unfortunate, since two dimensional displays can greatly simplify controls and improve the man machine interface in many cases. This article attempts to define the problem and offer some solutions.

Consider the display and input requirements of a simple mailing list system. For each record on the file there are approximately ten data fields. Each record is uniquely identified by a key which is arbitrarily assigned or is based upon some of the data fields in that record. A one-dimensional approach to updating an existing entry on the file would involve an interactive process of asking if each field in turn requires modification, then requesting input for the selected fields. Another one-dimensional approach would require the entry of a code word (such as "NAME") before a field to be updated. A two-dimensional approach to the problem would involve the use of a formatted screen. All ten fields would be displayed and updates could be keyed over the fields to be changed. This is far better than human engineering than in the one-dimensional case.

There are several alternative hardware approaches to the problem of two-dimensional I/O, which may be summarized as follows:

1. ignore the problem,
2. use memory mapped video displays and keyboard inputs,
3. use "smart" or "intelligent" terminals which support formatted screens,
4. use "dumb" terminals with cursor control capabilities.

It may also be properly used in those cases in which the benefits of two dimensional display do not outweigh the additional programming time and effort usually required compared to one dimensional of display and input.

The use of the memory mapped video displays is increasing. Most of the "appliance" micro computers (such as TRS 80, APPLE II, PET, TI, etc.) use this technique. Because the display represents the contents of an area of directly addressable memory, changes to the screen may be made very quickly. Since the micro processor has better control over this type of display, special effects, such as multiple cursors, multiple independent subscreens, graphics, colors, blinking, multiple intensity levels, protected fields, etc. are simpler to implement in displays. For two-dimensional display purposes memory mapped video displays are generally superior to terminals, and may be less expensive in many cases.

"Smart" or "intelligent" terminals which support formatted screens are used quite heavily on main computer systems. Until recently, the cost of these terminals was prohibitive for micro computer systems. There exists very little support in micro computer programming systems for the specialized data formats required by many of these terminals for proper operations. The screen formatting capabilities of many of these terminals are being ignored for these reasons, and they are being used as "dumb" terminals.

Most of the non "appliance" computers are connected to terminals. [illegible] of these terminals [illegible] computer controlled cursor movement and screen clear, and full duplex (non echo) mode, along with normal terminal operations. With proper programming, these terminals can be made to support two dimensional displays. As with "smart" terminals, the control functions are almost always performed with imbedded characters in the data stream. Full duplex operation is required so that the computer can control the display properly.

The primary problem impeding increased use of full screen displays on micro computers seems to be lack of programming support. Let us analyze the problem of generating and controlling any two dimensional display, in a high-level manner independent of hardware details.

Considering the top left corner of the display as row 1 and column 1, a field on the display may be defined by its screen number, position (column, row) length, and contents. This is the two dimensional analogue of the one dimensional case. The utility and power of two dimensional is considerably enhanced through what may be considered as field attributes. These attributes may include the following (and others, for particular displays):

- numeric only input,
- prohibited input (protected),
- intensity,
- reverse display,
- blinking,
- colors,
- underlining,
- etc.

In both one and two dimensional cases, the maximum byte of a given display must be considered.

Considering the two dimensional display as a group of fields (each defined by its position, length, contents and attributes), develop a data structure which allows the definition of a high level programming aid. This aid may probably be best implemented as a set of subroutines under an existing programming system. Using this technique, fields can be defined in a table. Then the subroutine can scan the table, and, using lower level device handlers and primitive functions specific to the display hardware, can control the display and input process. Except for feature selection in the field attributes and maximum byte dimensions of the display, the construction of the field table is independent of the display hardware. The input and output character strings which comprise most of the hardware dependencies may be assigned values in an initialization routine.

The same field table which provides the formats for the display can be used to provide the formats for the inputs, if any, to be made to the display. Input is allowed to any field not protected. Numeric only fields are those in which the input is restricted to the digits, plus, minus, comma, and period. The data input to a field replaces the previous data on the display and in the field table. A cursor usually designates the position on the display which will be modified if a displayable character is struck on the keyboard. Input into a protected or undefined field is not allowed. The cursor is normally initially positioned to the beginning of the first unprotected field on the display. If the current characters on the display is not to be modified, one of the following special functions may be selected:

- cursor right,
- cursor left,
- next field,
- previous field,
- current field,
- restore current field,
- first field,
- erase to end of field,
- insert character
- delete character,
- transmit.

On many terminals, some or all of these functions may require the simultaneous depression of two keys. Whenever a field is completed, a common field level edit subroutine is called to allow detailed, interactive editing of such fields as time, calendar dates, phone numbers, social security numbers, zip codes, etc.

The device handlers perform the low level hardware dependent functions necessary to support the higher level display and input functions. For memory mapped displays, the device handlers store or retrieve data to or from the display memory, possible after coordination with horizontal and vertical retrace signals. For all types of displays, the device handlers perform the elementary input (without echo) and output operations. Because of the large number of characters on the screen modified in many cases, there will usually be a perceptible screen update time. This can be minimized for serial terminals by setting the transfer rate as high as possible.

The primitive functions use the device handlers to perform specific, elementary functions which are also hardware dependent. The following list of functions is normally performed at this level:

- set display attributes,
- reset display attributes,
- output string of characters,
- input one character,
- clear screen,
- sound alarm,
- cursor left N position,
- cursor right N positions,
- cursor up N positions,
- cursor down N position,
- set cursor to (column, row).

Particular hardware requirements may lengthen this list.

Two high level subroutines (screen display and screen input) are defined. They use the field tables, primitive functions, and program logic to accomplish their respective tasks. In order that a given program might have more than one screen format defined, the high level subroutines process only those fields with screen number within an externally defined range. In a data entry application, the protected fields could be defined for a different screen number from the unprotected fields. Then the protected fields would be displayed only once and would remain on the screen, even after the unprotected fields have been reset. Other switches provide for automatic erasing of unprotected fields before input, automatic skipping to next field, and input only (error correction) operations.

Since the order of entry into the fields on the screen is controlled by the order of the corresponding field table entries, columnar or non sequential order of input is simple to implement. On virtually all "smart" terminals, the order is left to right, top to bottom. The field level edit capability of the input subroutine also allows for instant price extensions, conditional protecting and unprotecting of fields, and other high interactive features present in only the most "intelligent" terminals.

The operations of the high level screen display function may be summarized (in a pseudo language) as follows:

```
       I=-1
 LOOP: I=I+1
       IF NO MORE FIELDS, GO TO EXIT
       IF SCREEN (I) OUT OF RANGE, GO TO LOOP
       GET FIELD (I) CONTENTS FROM TABLE
       IF (ERASE ALL UNPROTECTED OR CLEAR FIELD BEFORE DISPLAY) AND UNPROTECTED, CLEAR
          FIELD (I) CONTENTS
       IF REQUESTED FOR FIELD (I), CLEAR SCREEN
       SET FIELD (I) ATTRIBUTES
       SET CURSOR TO (COLUMN (I), ROW (I))
       TRANSMIT FIELD (I) CONTENTS TO DISPLAY
       RESET FIELD ATTRIBUTES FOR FIELD (I)
       GO TO LOOP
 EXIT: RETURN
```

The operations of the high level screen input function may be summarized (in a pseudo language) as follows:

```
       CALL SCREEN DISPLAY FUNCTION
 RESET:I=-1
       DIRECTION=+1
 LOOP: I=I+DIRECTION
       IF I IS NEGATIVE, GO TO RESET
       IF NO MORE FIELDS, DIRECTION=-1, GO TO LOOP
       IF SCREEN (I) OUT OF RANGE, GO TO LOOP
       IF FIELD (I) PROTECTED, GO TO LOOP
       GET FIELD (I) CONTENTS FROM TABLE
       SET FIELD (I) ATTRIBUTES
       SET CURSOR TO (COLUMN (I), ROW (I))
```

5

```
         RESET FIELD POSITION POINTER
 INPUT:  INPUT A CHARACTER
         IF CHARACTER NOT CONTROL, GO TO NONCTL
         IF END OF FIELD INPUT, GO TO EOFIELD
         PROCESS CURSOR LEFT, CURRENT FIELD, NEXT FIELD, FIRST FIELD, PREVIOUS FIELD,
             RESTORE CURRENT FIELD INPUT REQUESTS
         IF BEYOND END OF FIELD, SOUND ALARM, GO TO INPUT
         PROCESS CURSOR RIGHT, ERASE TO END OF FIELD, INSERT CHARACTER, DELETE CHARACTER
             INPUT REQUESTS
         IF INVALID CONTROL CHARACTER, SOUND ALARM, GO TO INPUT
 NONCTL: IF BEYOND END OF FIELD, SOUND ALARM, GO TO INPUT
         IF NUMERIC INPUT REQUESTED FOR FIELD (I), IF INPUT CHARACTER NOT NUMERIC OR PLUS
             OR MINUS OR PERIOD SOUND ALARM, GO TO INPUT
         REPLACE FIELD (I) CHARACTER WITH INPUT CHARACTER
         INCREMENT FIELD POINTER
         IF NOT BEYOND END-OF-FIELD OR NOT AUTOSKIP, GO TO LOOP ELSE CHARACTER=NEXT-FIELD
 EOFIELD: UPDATE FIELD TABLE FOR FIELD (I) CONTENTS
         CALL FIELD EDITOR
         IF ERROR RETURN, SOUND ALARM, RESET TO FIELD, GO TO INPUT
         IF NEXT FIELD, DIRECTION=1, GO TO LOOP
         IF PREVIOUS FIELD, DIRECTION=-1, GO TO LOOP
         IF HOME CURSOR, GO TO RESET
         RETURN
```

Note that "smart" & "intelligent" terminals are treated as almost "dumb" terminals by these subroutines. Alternative formulations are possible in specific cases which make better use of particular terminal characteristics, but also limit the generality of the subroutines.

Once programming support for full screen display and input (in the form of high level subroutines, properly interfaced) has been developed, many types of application areas become much simpler to implement and to use. These include the following major areas:

data entry,
menus,
word processing,
text processing,
data base entry and update,
specialized applications,
games,
etc.

This article has attempted to present a programming methodology for attacking and simplifying the problem of full screen (two dimensional) display and input on micro computers. The method involves a high level, table driven, hardware independent, set of subroutines supported by lower level, hardware dependent, primitive functions and device handlers.

September 6, 1980.
346 Evans Road.
Nashville, Tennessee 37204.

Mr. Don Williams, Sr.
'68' Micro Journal.
3018 Hamill Road.
P. O. Box 849.
Hixson, Tennessee 37343.

Dear Sir:

This morning I dropped into one of the local Radio Shack stores and picked up a copy of the new TRS-80 Computer Catalog. I'd read rumors in '68' Micro Journal and other magazines about Radio Shack going to the 6800 or not going to the 6809, and I was interested in seeing what they'd done. The TRS-80 Color Computer and TRS-80 Videotex smart terminal use a 6809 and a 6847; the other TRS-80 models still use the Z-80. The TRS-80 Color Computer appears to be Radio Shack's reply to the Atari, APF and Texas Instruments home computers. The TRS-80 Videotex terminal also comes with an hour on the CompuServe timesharing system include . Neither machine has shown up in Nashville yet, but the specifications look good.

The TRS-80 Color Computer is built around the 6809E microprocessor and the 6847 video display generator. It comes with either 4K or 16K of dynamic memory and either 8K BASIC or 16K BASIC in ROM. Also available are a number of Program Paks of machine language programs in ROM, a color receiver, printer, cassette recorder, telephone modem and joysticks. A disk system may be available soon. Built-in TRS-80 Color Computer hardware also include a real time clock and RS-232 serial interface. The currently available Program Paks include Personal Finance, Space Commander, Football, Checkers, Chess, Music, Bingo Math, Pinball, and a diagnostic ROM.

The 8K Color Basic has strings, 9 place floating point arithmetic, ABS, INT, RND, SGN, SIN, SQR, PEEK, POKE, USR, JOYSTK (read joysticks), SET (select color) and SOUND (music through the TV speaker). The 16K BASIC has in addition TIME$ (read real time clock), PRINT$ (print in dollars and cents format), stronger graphics capability, program editing functions, better error messages, user defined keys and string arrays. The graphics commands compare favorably with the Apple II Hi-Res routines and include size changes of the displayed shape, line drawing function, shape rotation, shape movement, circle drawing function, rectangle drawing function, plus save and load graphic shapes.

The TRS-80 Videotex is a terminal version of the TRS-80 color computer and, judging from the catalog illustrations, has graphics capability. It comes with 4K RAM buffer, Videotex smart terminal software (also available for the Color Computer, other TRS-80 models, and the Apple II), and a direct connect modem. The best feature is a free hour on the CompuServe timesharing network. Radio Shack plans to, or has established its own bulletin service on CompuServe. It appears that electronic mail is coming closer and closer.

None of this hardware has appeared in Nashville yet. Probably we won't see a lot of it until the Christmas shopping season is here. Once we see the stuff for real, we can have a much better idea about what's what.

Very truly yours,

William M. Gambles

Dear friends,

After the introduction of the MC6809, having read some of the documentation on this subject, it didn't take much time to make me enthusiastic about this powerfull processor. As an owner of an ITT 2020 personal computer (the European second source for the APPLE II) soon the idea arose to expand my computer with the power of the 6809. Others have already done the same with a Z80 CPU. I am still brainstorming and I have probably not overlooked all the possibilities and/or problems. I would be glad with all the help I can get. At the same time I hope that my letter stimulates many hobbyists and proffesionals. If you have any suggestions or ideas, please send them to:

Jack v.d Watering
Nieuwe Molenweg 31a
4661 SH HALSTEREN
The Netherlands

STAR-KITS

P O Box 209
MT. KISCO, NEW YORK 10549

NEWS RELEASE

The lowest-cost single-board computer system yet has just been introduced by Star-Kits, P.O. Box 209, Mt. Kisco NY 10549. The Model SBC-02 computer is a minimal 4-chip system on a 4"x6" printed circuit board, which features a 6802 processor with 128 bytes of RAM, 2K of EPROM, and parallel or serial I/O. A wire-wrap area is provided for custom interfacing or other expansion. Because of its low cost, the SBC-02 is ideal for system development, limited-run production, and school or company training programs.

In single quantities, the computer costs $25 for a bare board with instructions, $75 for a parallel I/O kit, or $150 when wired and tested. Optionally, a machine level monitor called SUMBUG can be installed to provide program entry and control, single-stepping, breakpoints, and other front-panel functions from a serial terminal. SUMBUG runs on the 4-chip system, and is supplied separately in 2716 EPROM for $40 (included at no charge in the kit and wired versions.)

Additional support includes extensive tutorial literature, 4K floating point Basic in ROM which runs on the system, a cross-assembler for 6802 program development, SUMBUG ROMs for other 6800 systems, and consulting services.

07Sep80
4229 Estates Ct
Allison Park, PA. 15101

Don Williams, Sr
68' Micro Journal
P.O. Box 849
Hixson, TN 37343

Sometime I hope to own a "daisy wheel" printer and your servicing experience was of much interest to me.

Right now I am using a Teletype Model 43. You may be interested in the "problem" I had when I needed some work to be done on it. The machine "went dead" on a Thursday evening. Friday morning I called the service branch in Pittsburgh and was told to bring the unit in at my convenience, which I did later the same day. When I left it they were almost apologetic in saying that they doubted it would be ready before the following Monday morning. Early Monday morning I received a call telling me that it was ready. Replacement of a limit switch plus cleaning and oiling cost me $56, billed later.

In the future, whenever I buy major new gear, adequate servicing facilities will be high on my list of priorities.

T.G.Beckwith

LSI ENTERPRISES LTD

August 8, 1980

New Product Release: SIM80

LSI Enterprises is pleased to announce its newest product, SIM80. SIM80 is an 8080/8085 simulator that executes 8080 machine language programs directly without modifications. It is an "interpreter" written in 6809 and 6800 machine code. It is approximately 2 - 10 times slower than an 8080 running at similar clock speeds. It executes all of the 8080 code.

SIM80 also allows the user to call 6800 or 6809 code for applications requiring tight timing or for servicing I/O drivers in real time. This is done by utilizing an unused 8080 opcode.

Interfacing SIM80 is also simple. All that is required of the user is to supply three vectors for console input, output and monitor or operating system entry.

SIM80 is available on KC tape or Percom diskette for $34.95. It will also be available on Flex 2 diskette on November 1, 1980 for the same price.

May 20,1980

'68 Micro Journal
3018 Hamill Rd.
Hixson, Tenn. 37343

Dear Mr. Williams:

I am writing this letter to applaud a little known supplier of SS-50 boards, Walter Wimberly, 2914 Sunrise Dr., Orlando, Fla. 32803. Walt offers a 16K EPROM and now a 16K RAM card for this bus. Each has complete address decoding and so can be addressed into the upper 32k of memory.

I first purchased the EPROM board from Walt and was pleasantly surprised by its quality and documentation. This board is divided into four 4K byte blocks each independently addressable on any 4K boundary. It takes 2708's, 1K by 8 multi-voltage EPROMS, addressable anywhere in memory that isn't previously occupied. I use it, among other things, for my modified version of SWTBUG at $E000.

I just found out about and have bought Walt's most recent offering, an SS-50, 16K RAM card which uses 2114's. I am delighted. This card is organized as two 8K blocks each independently addressable to an unused 8K boundary anywhere within a 64K system.

The quality of the design and the board itself are excellent and I, for one, am very pleased with my investment.

This letter is intended to "spread the word." We have another source of SS-50 boards. There's another good guy with good products and I want to point him out.

Sincerely,

Dexter S. French, Jr.
142 Stoney Ridge Dr.
Longwood, Fla. 32750

TRANSFER SWTP 6800 COMPONENTS TO SWTP 69/K

Dan Grostick
21555 Plum Rd
Boca Raton, Fl 33433

I have been successful in transferring my 6800 based I/O and memory components to the new SWTP 69/K mainframe running FLEX 9 Version 2.6. I want to share my experiences with others considering this move.

OVERVIEW

I had been running FLEX on a SWTP 6800 with 40K of memory, a MP-L parallel interface, a MP-S serial interface, a MP-R eprom programmer, and a DC-2 mini-floppy controller.

There are two significant changes that should be noted when transferring any I/O device to the 69/K with the new MP-B3 motherboard. Each port now has 16 addresses as compared to 4 addresses in the 6800 MP-B and MP-B2 motherboards. This is accomplished by dedicating the UD3 and UD4 as the additional addressing lines (now RS2 & RS3 respectively). Also the +,- 12 volt inputs now supply +,- 16 volts.

MP-L, MP-S, MP-R, DC-2

The MP-L, MP-S and MP-R will plug into the slots without modification. The MP-S does use the 16 volt input but is not affected by the increased voltage. The DC-2 mini-floppy controller is not directly compatible with the new motherboard as it was designed to be decoded on port 5 & 6 on the 6800 motherboard.

A relatively simple modification requiring one 7400 IC and a few wires can be easily installed to provide the necessary decoding. First two lands must be cut. Position the DC-2 controller bottom side up with the J2 30 pin connector towards you. Cut the leftmost land (UD3 - the CS5 input) between J2 and the first feedthru. Next turn the board over and locate the leftmost land on the top side (I/O port select) and follow it to the first feedthru. Turn the board over and cut the I/O select line at the short land located between R12 and D2. Mount the IC (a 7400) piggyback on IC11 (bending up all legs except pins 7 & 14 which should be soldered to IC11). Connect the new IC as shown in Figure 1.

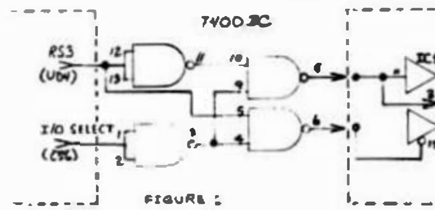


FIGURE 1

DAN GROSTICK 305-[illegible](H)		
21565 PLUM RD 305-[illegible](W)		
BOCA RATON FL 33433	6/27/80	
SIZE A	DRAWING NO. DC-2 MODIFICATIONS	REV —
	SHEET 5 OF 5	LTR

Note that the inputs are CS6 (I/O port select) and RS3 (MP-B2 UD4), not the previously used UD3. The outputs from the chip may be connected to the feedthrus inboard from the cut lands. Double check your wiring and plug the modified controller into port 1 as required by FLEX 9. Note that the DC2 board uses zener diodes to regulate inputs from the +,- 16 volt supply and is not affected by the increased voltages from the new power supply.

MEMORY

The SWTP MP-MP 4K memory, MP-8M 8K memory, and the Digital Research 16K static memory cards do not require any special modifications for the 69/K.

69/K GENERAL COMMENTS

SWTP should be complimented on a well designed and implemented package that allows a reasonable level of compatibility with previous products. The most significant shortcoming is a very noisy cooling fan that drowns out any chance of silence. S-BUG has an improved user interface as compared to the previous monitors. FLEX 9 Version 2.6, although lacking any multitask capability, offers an excellent price / performance ratio.

CONCLUSION

Transfering existing components to a new system is a cost effective way to upgrade without losing existing capability. In consideration of the economics, the move to a 6809 system did not cost much more than upgrading with a MP-09 processor card. The sale of the 6800 system plus the cost of a MP-09 was close enough to the cost of the new system to justify the move.

volset

When your disk library gets as large as mine with nearly a hundred mini-floppies, there is often a need to change the volume names and or serial numbers of disks without having to reinitialize them. VOLSET was written to do just this.

FLEX disks have a System Information Record which contains information concerning disk space for FLEX useage and the volume name and serial number. The volume name and serial are not used by FLEX at all so changing them is quite safe.

There is a FLEX function to open and read the SIR but its method of use is not entirely clear from the documentation so I chose to do single sector I/O directly on the SIR.

The actual makeup of the SIR is in fact best described in the 6809 adaption guide. The offset for the volume name is 16 decimal bytes and the serial number 27 bytes. The volume name is in ASCII but the number is in BINARY. The number therefore has a maximum value of 32767. My system has two drives and I use drive 1 as the work disk all of the time. I fixed the drive for VOLSET to 1 for this reason and also I think that changing the system disk SIR could, under certain situations eg spooling, be dangerous.

The command format is:

VOLSET,(NEWNAME),(NEWNUMBER).

The fields NEWNAME and NEWNUMBER are optional and if either field is invalid then no changes will be made to that field. The actual program and the listings are sufficient for the description of operation. The conversion to FLEX09 is just a matter of changing a few addresses and submitting the source to the TSC 6809 assembler which will automatically convert 6800 instructions to 6809 equivalents in flight.

Finally, I recently reassembled the PROT command to allow files to be set to read protect and also displayed as having read protect on. The problem is that even if the read protect bit (bit 5, byte 15 in FCB) is on then FLEX ignores it. I have not discovered the exact mechanism employed by FLEX to check for read protected files, but if the instruction at location $B9C3 is replaced by 2 NOP's then the file will be checked for read protect and the appropriate error message reported. The problem with this patch is that once a file is read protected then the PROT command will not allow the status to be changed as the file is read protected. The PROT command could be made to temporarily alter $B9C3 to $2706 to enable read protection to be removed. There are a number of tests and switches in FLEX which appear to be connected with file protection one of these is $B41A which is tested before the read protect test is made.

```
                        NAM     VOLSET

                ****************************************************
                * THIS PROGRAM WILL ALLOW THE USER TO ALTER        *
                * THE DISK NAME AND OR NUMBER WITHOUT HAVING       *
                * TO REINITIALIZE THE WHOLE DISK.                  *
                *                                                  *
                * COMMAND FORMAT : VOLSET,[NEWNAME],[NEWNUMBER]    *
                * IF NEWNAME OR NEWNUMBER ARE NULL THEN THEY       *
                * WILL NOT BE CHANGED.                             *
                *                                                  *
                * DRIVE NUMBER IS FORCED TO 1 AS CHANGING THE SIR  *
                * ON DRIVE 0 IS RISKY UNDER CERTAIN CONDITIONS.    *
                *                                                  *
                * AUTHOR   : DAVID BOABBY                          *
                * WRITTEN  : 26/8/80                               *
                * UPDATED  :                                       *
                ****************************************************

                *
                * SYSTEM EQUATES
                *
A100            CODE    EQU     $A100      UTILITY AREA
A840            FCB     EQU     $A840      SYSTEM FCB
                * CONSTANTS
0001            DRNUM   EQU     1          FORCE DRIVE 1
0009            READS   EQU     $9         READ SINGLE SECTOR
000A            WRITES  EQU     $0A        WRITE SINGLE SECTOR
                * FCB LOCATIONS
0000            FUNCT   EQU     $0         FUNCTION CODE HERE
0001            ERROR   EQU     $1         ERROR BYTE
0003            DRIVE   EQU     $3         DRIVE NUMBER 0-3
001E            TRACK   EQU     30         CURRENT TRACK
001F            SECTOR  EQU     31         CURRENT SECTOR
A880            SECBUF  EQU     FCB+$40    DATA PORTION OF FCB
                * SYSTEM INFORMATION RECORD OFFSETS
0010            VOLNAM  EQU     16         VOLUME NAME (11 CHARS)
001B            VOLNUM  EQU     27         VOLUME NUMBER IN BINARY (2 BYTES)
                * OFFSETS IN FCB
A89B            NUMS    EQU     SECBUF+VOLNUM LOCATION OF VOLUME NO.
A890            NAMS    EQU     SECBUF+VOLNAM LOCATION OF VOLUME NAME
                * FLEX SUBROUTINES
AD03            WARMS   EQU     $AD03      WARM START FLEX
B406            FMS     EQU     $B406      FLEX E TRY POINT
AD48            INDEC   EQU     $AD48      GET DECIMAL NO IN X
AD2D            GETFIL  EQU     $AD2D      GET FILE SPEC
                *
                * MAIN PROGRAM
                *
A100                    ORG     CODE
A100 20 15      START   BRA     START1
A102 01         VERS    FCB     1          VERSION NUMBER
A103            XTEMP   RMB     2          TEMP SAVE AREAS
A105            XTEMP1  RMB     2
A107 00         FLAG    FCB     0          UPDATE FLAG
                *
                * PSEUDO FCB FOR GETFIL
                *
A108            PFCB    RMB     4
A10C            PFNAM   RMB     8          FILE NAME PUT HERE
A114 00                 FCB     0,0,0      CLEAR EXTENSION FIELD
A115 00 00
                *
A117 CE A8 40   START1  LDX     #FCB       GET SIR FIRST

A11A 86 09              LDA A   #READS     SS READ OP CODE
A11C A7 00              STA A   FUNCT,X
A11E 86 01              LDA A   #DRNUM     FORCE DRIVE NUMBER
A120 A7 03              STA A   DRIVE,X
A122 86 00              LDA A   #0         TRACK 0
A124 A7 1E              STA A   TRACK,X
A126 86 03              LDA A   #3         SECTOR 3
A128 A7 1F              STA A   SECTOR,X
A12A BD B4 06           JSR     FMS        GET SECTOR
A12D CE A8 90           LDX     #NAMS      NAME FIELD IN SIR
A130 FF A1 05           STX     XTEMP1
                * READ NEW VOLUME NAME
                * IF VOLUME NAME IS A NULL ENTRY OR IT IS INVALID
                * THEN IT WILL NOT BE CHANGED.
A133 CE A1 08           LDX     #PFCB      PSEUDO FCB.
A136 BD AD 2D           JSR     GETFIL     GET NEW NAME
A139 25 1D              BCS     CHGNUM     WAS NAME OK ?
A13B CE A1 0C           LDX     #PFNAM     YES IT WAS
                * MOVE NEW NAME INTO SIR FROM PSEUDO FCB
A13E C6 0B              LDA B   #11        NO OF CHARS IN NAME
A140 A6 00      PLOOP1  LDA A   0,X        GET CHAR
A142 08                 INX                POINT TO NEXT
A143 FF A1 03           STX     XTEMP
A146 FE A1 05           LDX     XTEMP1
A149 A7 00              STA A   0,X        PUT NAME INTO SIR
A14B 08                 INX
A14C FF A1 05           STX     XTEMP1     POINT TO NEXT
A14F FE A1 03           LDX     XTEMP      GET SOURCE FIELD
A152 5A                 DEC B              BUMP COUNT
A153 26 EB              BNE     PLOOP1     LOOP TILL DONE
A155 7C A1 07           INC     FLAG       SET UPDATE FLAG
                * CHANGE DRIVE NUMBER.
                * IF NEW VOLUME NUMBER IS NEITHER VALID NOR
                * PRESENT THEN NUMBER IS UNCHANGED.
A158 BD AD 48   CHGNUM  JSR     INDEC      GET NEW VOLUME NUMBER
A15B 25 09              BCS     FINISH     NUMBER NOT VALID
A15D 5D                 TST B              CHECK FOR SEPERATOR
```

```
A15E 27 06                BEQ   FINISH
A160 FF A8 9B             STX   NUM     NEW NUMBER
A163 7C A1 07             INC   FLAG    SET UPDATE FLAG
                  * WRITE OUT NEW SIR IF CHANGE FLAG IS NON ZERO
A166 7D A1 07    FINISH   TST   FLAG    SEE IF WRITE IS REQUIRED
A169 27 0A                BEQ   NOACT   DONT BOTHER
A16B 86 0A                LDA A #WRITES SET FCB TO WRITE
A16D CE A8 40             LDX   #FCB
A170 A7 00                STA A FUNCT,X
A172 BD B4 06             JSR   FMS     WRITE OUT UPDATED SIR
A175 7E AD 03    NOACT    JMP   WARMS   THATS IT
                          END   START
```

NO ERROR(S) DETECTED

SYMBOL TABLE:

```
CHGNUM A158   CODE   A106   DRIVE  0003   DRNUM  0001   ERROR  0001
FCB    A840   FINISH A166   FLAG   A107   FMS    B406   FUNCT  0000
GETFIL AD2D   INDEC  AD48   NAMS   A890   NOACT  A175   NUMF   A89B
PTCB   A108   PFNAM  A10C   PLOOP1 A140   READS  0009   SECBUF A880
SECTOR 001F   START  A100   START1 A117   TRACK  001E   VERS   A102
VOLNAM 0010   VOLNUM 001B   WARMS  AD03   WRITES 000A   XTEMP  A103
XTEMP1 A105
```

The attached program, PULSE2.TXT, may be of interest to some readers who are all thumbs where hardware is concerned.

The program uses the CB2 line of a 6820 PIA to generate some pulses. The length of time the output is high or low is variable and selectable. The main work section is LOOP. It operates by setting bit 3 high and then low (lines 56 & 61). The length of time CB2 is on or off is determined by the HEX values you select during SETUP. As you observe the waveform on a 'scope you may reset the variables if you enter the ESC key.

HELP - HELP - HELP - HELP - HELP

Problem with 000 ASMB.S - Only works if Option 8 (no creation of binary file) is selected. Error codes generated 0100, 08 and sometimes 0303! Binary File is logged in directory but with 0000 as beginning and ending track and sector!
Please send me your solution in 100 bytes or less. Thanks.

Gene Embry
Route 1 Box 151-M
Morrisville, NC 27560

PULSE2.TXT   SSB MNEMONIC ASSEMBLER   PAGE 1

```
                 1:          NAM   PULSE2.TXT
                 2:
                 3: * This will use CB2 to generate pulses.
                 4:
B003             5: CRB      EQU   $B003
B002             6: DDRB     EQU   CRB-1
B002             7: DATA     EQU   DDRB
                 8: *
                 9: *    SB-DOS equates
                10: *
D2DC            11: ZCRLF    EQU   $D2DC
D2A6            12: ZOUTST   EQU   $D2A6
D209            13: ZINCH    EQU   $D209
                14: *
                15: *  SWIBUG equates
                16: *
E055            17: BYTE     EQU   $E055
E047            18: BADDR    EQU   $E047
                19: *
                20: * Program constants
                21: *
001B            22: ESC      EQU   $1B           stop the generator
B004            23: CNTRL    EQU   $B004
                24: *
                25: *
B000            26:          ORG   $B000
                27: *
B000 20 04      28:          BRA   INIT
B002            29: TIMEON   RMB   1
B003            30: TIMEOF   RMB   1
B004            31: MIN      RMB   2
                32: *
B006 B6 B003    33: INIT     LDA A CRB
B009 84 FB      34:          AND A #$FB
B00B B7 B003    35:          STA A CRB
B00E 86 34      36:          LDA A #%00110100  configuration
B010 B7 B003    37:          STA A CRB
                38: *
B013 BD D2DC    39: SETUP    JSR   ZCRLF
B016 CE B078    40:          LDX   #MESON
B019 BD D2A6    41:          JSR   ZOUTST
B01C BD E055    42:          JSR   BYTE
B01F B7 B002    43:          STA A TIMEON
B022 BD D2DC    44:          JSR   ZCRLF
B025 CE B092    45:          LDX   #MESOFF
B028 BD D2A6    46:          JSR   ZOUTST
B02B BD E055    47:          JSR   BYTE
B02E B7 B003    48:          STA A TIMEOF
B031 BD D2DC    49:          JSR   ZCRLF
B034 CE B0B0    50:          LDX   #DURAT
B037 BD D2A6    51:          JSR   ZOUTST
B03A BD E047    52:          JSR   BADDR
B03D FF B004    53:          STX   MIN
B040 BD D2DC    54:          JSR   ZCRLF
B043 B6 B003    55: LOOP     LDA A CRB
B046 8A 08      56:          ORA A #%00001000  make level one
B048 B7 B003    57:          STA A CRB
B04B F6 B002    58:          LDA B TIMEON
B04E 8D 1E      59:          BSR   MILDLY
B050 B6 B003    60:          LDA A CRB
B053 84 F7      61:          AND A #%11110111  make level zero
B055 B7 B003    62:          STA A CRB
B058 F6 B003    63:          LDA B TIMEOF
B05B 8D 11      64:          BSR   MILDLY
B05D B6 B004    65:          LDA A CNTRL       check for a break
B060 44         66:          LSR A
B061 24 08      67:          BCC   AGAIN       no break from keyboard A 1A
B063 B6 B005    68:          LDA A CNTRL+1     get the input
B066 81 1B      69:          CMP A #ESC
B068 26 02      70:          BNE   AGAIN
B06A 20 A7      71:          BRA   SETUP
B06C 20 D5      72: AGAIN    BRA   LOOP
                73: *
B06E FE B004    74: MILDLY   LDX   MIN
B071 09         75: DEL1     DEX
B072 26 FD      76:          BNE   DEL1
B074 5A         77:          DEC B
B075 26 F7      78:          BNE   MILDLY
B077 39         79:          RTS
                80: *
                81: *
B078 54         82: MESON    FCC   /Time unit for +5 volts = /
B091 00         83:          FCB   0
B092 54         84: MESOFF   FCC   /Time unit for ground state = /
B0AF 00         85:          FCB   0
B0B0 54         86: DURAT    FCC   /Time Expanding Factor = /
B0CB 00         87:          FCB   0
                88:          END
     NO ERROR(S) DETECTED
```

SYMBOL TABLE:

```
AGAIN  B06C   BADDR  E047   BYTE   E055   CNTRL  B004
CRB    B003   DATA   B002   DDRB   B002   DEL1   B071
DURAT  B0B0   ESC    001B   INIT   B006   LOOP   B043
MESOFF B092   MESON  B078   MILDLY B06E   MIN    B004
SETUP  B013   TIMEOF B003   TIMEON B002   ZCRLF  D2DC
ZINCH  D209   ZOUTST D2A6
```

D.V. GOADBY
2 LUPIN CLOSE
HINCKLEY LEICS
ENGLAND LE10 2UJ

715-1430 First St. East
Cornwall, Ontario
Canada K6H 6H1
Nov 30, 1980

68 Micro Journal
3018 Hamill Road
P.O. Box 849
Hixson, Tennessee
U.S.A 37343

Dear Sirs:

I am very satisfied with a product advertised in your magazine, and would like to pass my comments on to your readers.

DYNASOFT Systems PASCAL was shipped promptly and met all advertising claims. It is very compact and easily usable on a small system with only a single cassette. The manual is very good, but does assume some basic knowledge of PASCAL. Included is an interesting sample test program that runs well.

The whole PASCAL system resides in memory at once and consists of (from the bottom up) of the P-code interpreter, supervisor, editor and compiler. From there up is workspace. The editor allows you to write the source into the workspace, which can be saved after returning to the supervisor. The compiler can then be run to convert the source to P-code in the workspace. The supervisor can then start the execution of the user P-code.

A nice feature is the ability of the supervisor to move the user P-code down to the area occupied by the editor and compiler resulting in a much smaller 'run-time' package.

This PASCAL can manipulate 8 bit bytes and 16 bit integers in external memory (even I/O devices) by use of a POINTER type. This combined with the ability to call and pass parameters to machine language routines provides a lot of power. I also found that the P-code execution did not seem to mind running with interrupts enabled and handled by machine language routines.

All these features combined with the manual's information for adapting to any hardware configuration make this a handy package for small control applications, even if you already have a disc based PASCAL.

Maybe enough interest in Mr. Jost's software will encourage him to add REAL, about the only thing missing from this PASCAL.

Sincerely,

Eric Pierce C.E.T.

21 September 1980

68' Micro Journal
3018 Hamill Rd.
P.O. Box 849
Hixson, TN 37343

Dear sirs,

I have developed some software in BASIC which:

1) will print en masse to companies in DATA statements
   a) for a cover letter
   b) for an envelope

2) will search for a particular company
   a) for a cover letter
   b) for an envelope

3) will provide a complete listing of one's personal belongings entered in DATA statements

4) will search for an item, manufacturer or value greater then that entered

These programs provide hard copy to port 7 using MiniFlex BASIC.

While there is nothing unusual about these programs I would like to know if such programs can be copyrighted as I was thinking of selling them for $10.00 for a version on floppy. It might save someone hours writing the programs and just involve the keyboard slavery of entering DATA statements.

Does anyone know where to insert jump to routines in Dyna-Soft PASCAL for printer driver routines?

Thank you,

Jeffrey M Craig.

Jeffrey M. Craig
Apt. 912 - 3001 S. King Dr.
Chicago, IL 60616

HELP!

Does anyone know of the availability of a program, running under SSB 6809 DOS69, that will read FLEX files and convert them to SSB DOS format? Has someone figured out how to make the SSB TEXT PROCESSOR provide vertical page centering for letters?

John R Steele
10415 SW 115 Ct.
Miami, Florida 33176

CLASSIFIED

Immediate Openings for Qualified Programmers.

Call Toll Free- Control Systems, Inc./Microsystems Division for Mr. Dave Allen. 1-800-255-4411.

***

SWTPC 6800, 40K system including CT-82, MF-68, CT-30, SWTBUG, FLEX2.0, etc. $2000.

Sherman Wantz, Sebring, Florida, Phone: 813-385-4177.

***

Printer $250.00, TSC cassettes w/manuals, sell or trade for Z80 stuff.

Peter Hille, Box 196, Somerset, Ca 95684, Phone: 916-626-0749.

***

CT-64 with video mod Hitachi 12-in TV monitor. $280. CT-64 alone $200.

John J Gildewell, 3623 Charlene Dr., Dayton, OH 45432, Phone: 513-426-3867.

***

SMOKE SIGNAL BROADCASTING

Presents

3 Powerful New SS-50/SS-50C Boards

DCB-4
Disk Master
Double Density Controller Board and DOS68D Double Density DOS
$449.00

The new DCB-4 is a truly state-of-the-art development which allows up to 366K bytes to be stored on a single 5¼" disk and has these outstanding features:

- Up to four 5¼" and four 8" drives can be handled in the same system with a user definable logical unit table. (DOS68D will be compatible with future hard disk systems).
- Under software control, the user can select the following for any drive:
  - ☆ Single sided or double sided operation.
  - ☆ Single density or double density data.
  - ☆ 5¼" or 8".
  - ☆ Stepping Rate.
  - ☆ 40 track or 35 track density on double sided 5¼" drives.
  - ☆ User can select the system boot configuration.
- Occupies only 16 bytes of memory space (F760-F76F standard). User selectable to any 16 byte address space.
- Can read and write a single sector by itself. On-board buffer memory allows full interrupt capability in interrupt driven systems. Once data transfer has been initiated, no more processor time is required.
- Contains extended decoding circuitry for extended addressing per SS—50C bus which can be enabled by an option jumper.
- SS8 provides a means for copying software written by older versions of DOS68 to be read by DOS68D. All new media formatted by DOS68D can be read by all older versions of DOS68. DOS68 is SSB's 6800 disk operating system.
- Track Ø of side Ø is recorded in single density per IBM standard.
- Phase-locked-loop assures highest data integrity attainable.

All of these features are available for immediate delivery on one standard 5½" x 9" 50 pin SS—50/SS—50C card for only $449.00. The price includes DOS68D version 5.1, MONITOR object code on diskette, and a manual with the source listing.

SCB-69
Super Computer Board
6809CPU Board
$299.00

The most versatile 6809 CPU Board on the market is now available from Smoke Signal Broadcasting and has the following features:

- Standard 2 MHz operation. (Shipping 1.5 MHz until 68B09 available)
- 20 bit address generation for up to 1 Mbyte of memory. Uses an improved address translation RAM which is compatible with present extended addressing schemes yet requires much less overhead when used in multi-user systems.
- All on-board devices can be switch selected to occupy any or all extended pages. Any on-board device may be disabled and its memory space is then available for external memory.
- Standard real—time clock (time—of—day, day—of—week, day—of—month) with battery back up capable of generating programmable interrupts.
- Up to 20K of EPROM can be installed on the CPU Board.
- Standard 1K of RAM on board.
- Includes improved 6809 Monitor (and source listing).
- Contains an FPLA for decoding EPROM address and optional devices. Switches are used to select 2K/4K EPROM and Fast/Slow I/O.
- Contains provision for optional 9511/9512 floating point processor.
- NMI line is user selectable to work with either SS—50 or SS—50C busses.

Price for the new SCB—69 is only $299.00 for an assembled, burned-in fully tested board.

M-32-X
32K
Memory Board
$539.00 $439.00

The first and only 32K Static Ram Board on standard size (5½" x 9") SS—50/SS—50C Bus Circuit Card is made by Smoke Signal.

- Switch selectable to any 4K boundary.
- Any 4K block may be switch enabled or disabled.
- Fully compatible with SS—50C extended addressing (allows memory decoding up to 1 Mbyte).
- Extended addressing capability may be switched off for compatibility with SS—50 systems.
- Gold Bus Connectors for high reliability.
- Guaranteed 2MHz operation (tested at 2.2 MHz).
- Low power consumption — 8 volts at 2.4 amps typical.

M—32—X 32K Memory Board is priced at $539.00.

M—24—X 24K Memory Board expandable to 32K, is $439.00.

And our M—16—X 16K board is back to the old price of $299.00.

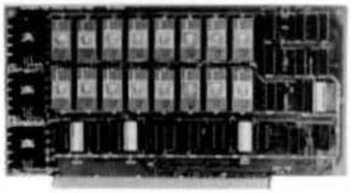
SMOKE SIGNAL BROADCASTING®

31336 Via Colinas, Westlake Village, CA 91361, (213) 889-9340

See GIMIX Ad Pages 3 & 56

DIGITAL RESEARCH COMPUTERS
(214) 271-3538

32K S-100 EPROM CARD
NEW!



$74.95
KIT

USES 2716's

Blank PC Board - $34

ASSEMBLED & TESTED
ADD $30

SPECIAL: 2716 EPROM's (450 NS) Are $14.95 EA. With Above Kit.

KIT FEATURES
1. Uses +5V only 2716 (2Kx8) EPROM's
2. Allows up to 32K of software on line!
3. IEEE S-100 Compatible
4. Addressable as two independent 16K blocks
5. Cromemco extended or Northstar bank select
6. On board wait state circuitry if needed
7. Any or all EPROM locations can be disabled
8. Double sided PC board, solder-masked, silk-screened
9. Gold plated contact fingers
10. Unselected EPROM's automatically powered down for low power
11. Fully buffered and bypassed
12. Easy and quick to assemble

16K DYNAMIC RAM PARTIALS

LOOK! INTEL 2108 8K X 1 RAMS LOOK!
8 FOR $9.95 32 FOR $35
FACTORY PRIME!

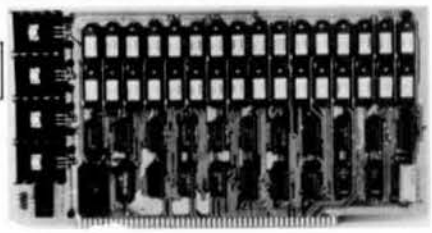
Huge special purchase of INTEL Dynamic RAM's. These are 2108-4, 300NS, 8K, Ceramic DIP. The 2108 is the INTEL 2116 (16K) tested for either upper or lower 8K only. These are factory prime. Full Spec. See INTEL 1978 Cat. for details or Memory Design Handbook for application data. Both IMSAI and EXTENSYS did mfg. S-100 RAM boards using these devices. — P.S. These devices will not work in the SD EPANDORAM™. Please specify upper or lower 8K. (S1626 or S1627). A supereasy RAM to interface to a Z80, 16 PIN DIP.

FOR 4MHZ — PRICE CUT! — LOW POWER - 300NS — 8 FOR $37.50
2114 RAM SALE!
4K STATIC RAMS. MAJOR BRAND, NEW PARTS.
These are the most sought after 2114's, LOW POWER and 300NS FAST.
8 FOR $37.50

16K STATIC RAM KIT-S 100 BUSS

PRICE CUT!
$199.95 KIT

FOR 4MHZ
ADD $10



KIT FEATURE
1. Addressable as four separate 4K Blocks
2. ON BOARD BANK SELECT circuitry (Cromemco Standard!). Allows up to 512K on line!
3. Uses 2114 (450NS) 4K Static Rams
4. ON BOARD SELECTABLE WAIT STATES
5. Double sided PC Board, with solder mask and silk screened layout. Gold plated contact fingers
6. All address and data lines fully buffered
7. Kit includes ALL parts and sockets
8. PHANTOM is jumpered to PIN 67
9. LOW POWER under 1.5 amps TYP. ALL from the +8 Volt Buss
10. Blank PC Board can be populated as any multiple of 4K.

BLANK PC BOARD W/DATA-$33
LOW PROFILE SOCKET SET-$12
SUPPORT IC'S & CAPS-$19.95
ASSEMBLED & TESTED-ADD $35

OUR #1 SELLING
RAM BOARD!

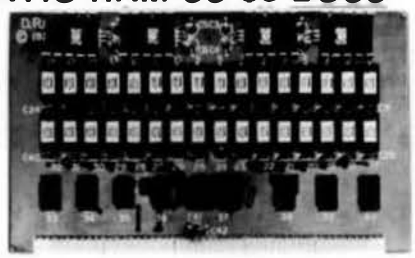
16K STATIC RAM SS-50 BUSS

PRICE CUT!

$210 KIT

FULLY STATIC!

FOR 2MHZ
ADD $10



FOR SWTPC
6800 BUSS!

ASSEMBLED AND
TESTED - $35

KIT FEATURES
1. Addressable on 16K Boundaries
2. Uses 2114 Static Ram
3. Fully Bypassed
4. Double sided PC Board. Solder mask and silk screened layout
5. All Parts and Sockets included
6. Low Power: Under 1.5 Amps Typical

BLANK PC BOARD—$30 COMPLETE SOCKET SET—$12
SUPPORT IC'S AND CAPS—$19.95

STEREO!
NEW! **S-100 SOUND COMPUTER BOARD** NEW!

At last, an S-100 Board that unleashes the full power of two unbelievable General Instruments AY3-8910 NMOS computer sound IC's. Allows you under total computer control to generate an infinite number of special sound effects for games or any other program. Sounds can be called in BASIC, ASSEMBLY LANGUAGE, etc.

COMPLETE KIT!
$84.95
(WITH DATA MANUAL)

BLANK PC
BOARD W/DATA
$31

KIT FEATURES:
- TWO GI SOUND COMPUTER IC'S
- FOUR PARALLEL I/O PORTS ON BOARD
- USE ON BOARD AUDIO AMPS OR YOUR STEREO
- ON BOARD PROTO TYPING AREA
- ALL SOCKETS PARTS AND HARDWARE ARE INCLUDED
- PC BOARD IS SOLDERMASKED, SILK SCREENED, WITH GOLD CONTACTS
- EASY, QUICK, AND FUN TO BUILD, WITH FULL INSTRUCTIONS
- USES PROGRAMMED I/O FOR MAXIMUM SYSTEM FLEXIBILITY

Both Basic and Assembly Language Programming examples are included

SOFTWARE

SCL™ is now available! Our Sound Command Language makes writing Sound Effects programs a SNAP! SCL™ also includes routines for Register-Examine-Modify, Memory-Examine-Modify, and Play-Memory. SCL™ is available on CP/M* compatible diskette of 2708 or 2716. Diskette - $24.95 2708 - $19.95 2716 - $29.95 Diskette includes the source. EPROM'S are ORG at E000H.

4K DYNAMIC RAM BLOWOUT!
SAME AS INTEL 2107B!
4K RAMS AT AN UNBELIEVABLE 50¢ EACH!!!

Prime, new, National Semi., 1979 date coded, full spec parts. N.S. #MM5280-5N. Same as INTEL 2107B-4, T.I. TMS4060, NECuPD411, etc. We bought a HUGE QTY. from a West Coast Distributor at truly DISTRESS PRICES! One of the most popular and reliable RAM's ever made. These parts have been used by almost all Major Computer Main Frame Mfg. the world over! Arranged as 4K x 1, 270 NS Access Time, 22 Pin DIP. These units DO NOT use multiplexed addressing, thus making REFRESH and other timing very simple. See INTEL MEMORY DESIGN HANDBOOK for full application notes. The NAT. SEMI. MEMORY DATA BOOK is available at most Radio Shack Stores. Prime units in original factory tubes!

(With Pin Out Data)
#5280-5N 4096 BITS x 1 270 NS ACCESS
8 FOR $4.95 32 FOR $16
FACTORY CASE (450 PCS) — $180
Sockets Special: 22 Pin Low Profile (With Purchase of 5280's) 8 FOR $1.

COMPUTER PARTS SPECIALS

74LS175 - .99	8035 Intel Single Chip CPU - 5.95
74LS240 - 1.79	Signetics 2901 4 Bit Slice - 6.95
74LS241 - 1.79	AMD 2903 4 Bit Super Slice - 12.50
74LS244 - 1.79	AMD 29705 Dual Port RAM - 8.95
74LS373 - 1.99	

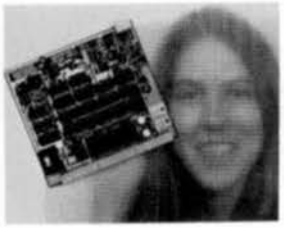
NEW! G.I. COMPUTER SOUND CHIP

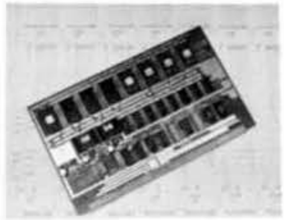
AY3-8910. As featured in July, 1979 BYTE! A fantastically powerful Sound & Music Generator. Perfect for use with any 8 Bit Microprocessor. Contains: 3 Tone Channels, Noise Generator, 3 Channels of Amplitude Control, 16 bit Envelope Period Control, 2-8 Bit Parallel I/O, 3 D to A Converters, plus much more! All in one 40 Pin DIP. Super easy interface to the S-100 or other busses. $11.95 PRICE CUT!
SPECIAL OFFER: ~~$14.95~~ each Add $3 for 60 page Data Manual.

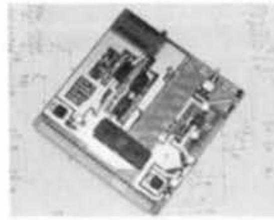
Digital Research Computers
(OF TEXAS)
P.O. BOX 401565 • GARLAND, TEXAS 75040 • (214) 271-3538

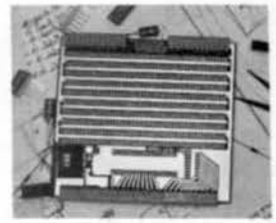
TERMS: Add $1.50 postage. We pay balance. Orders under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. 90 Day Money Back Guarantee on all items. Orders over $50, add 85¢ for insurance.

*TRADEMARK OF DIGITAL RESEARCH. NOT ASSOCIATED WITH DIGITAL RESEARCH OF CALIFORNIA, THE SUPPLIERS OF CPM SOFTWARE.


DS-68 DIGISECTOR


PSB-08 PROM SYSTEM BOARD


B-08 EPROM PROGRAMMER


UIO UNIVERSAL I/O BOARD

INNOVATIVE PRODUCTS FOR 6800 USERS

DS-68 DIGISECTOR is a random access video digitizer featuring 256 × 256 picture element scan and 64 levels of grey scale, with conversion times as low as 3 microseconds per pixel. It accepts either interlaced (NTSC) or non-interlaced (industrial) video input. Use it for computer portraiture, moving target indicators, precision security systems, fast to slow scan conversion... with clever software, the Digisector can read just about anything. Truly a professional tool at a price you can afford. $169.95

DM-85 DISK MIXER is an add-on board for the Smoke Signal Broadcasting BFD-68A Disk Controller which allows operation of both 8" and 5" drives. Controller mode (8" or 5") is selected on a drive-by-drive basis, so any mix of 5" and 8" drives is allowable. The 2" × 3" PC board mounts inconspicously on the back of the BFD-68A. Its operation is completely transparent to software. An oscilloscope is required for the setup procedure. Kit Price: $39.95

B-08 2708 EPROM PROGRAMMER is a compact unit that fits in the 6800's I/O slot. A safety switch and LED indicator provide control over the high programming voltage generated on board. An industrial quality Textool socket and extended board height allow effortless PROM insertion and retrieval. Fully commented source listings of U2708 is included in the Owner's Manual. $99.95

U2708 utility for testing, burning, verifying and copying 2708s in EPROM. $29.95

M6809 EMULATOR is a machine language program that will emulate all of the functions of the Motorola 6809 third generation microprocessor. Developed for use on any 6800 system, the program allows software development and debugging. The 3K byte program is complete with a 6809 mini-monitor and single-step trace routines. Fully commented source listing included. Specify Smoke Signal Broadcasting or FLEX™ disk, or KCS cassette. $49.95

PSB-08 PROM SYSTEM BOARD features 1K of high speed, low-power RAM and space for up to 8 2708 EPROMs, both DIP-switch addressable to start on any 8K boundary in memory. The exclusive I/O select feature allows you to move I/O locations up to any unused 1K block in the EPROM memory space. This permits memory expansion to a full 56K of contiguous user RAM. $119.95

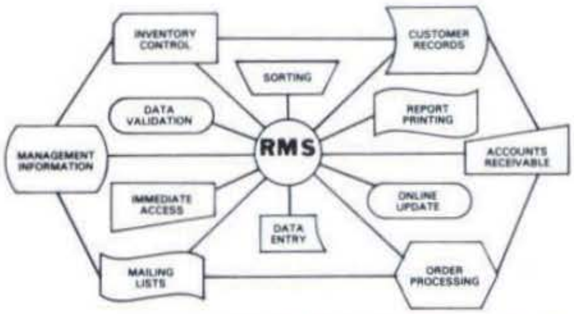
UIO UNIVERSAL I/O BOARD helps you with your custom interfaces. It has space for a 40-pin wire wrap socket into which you may plug any of Motorola's 40 or 24-pin interface chips. All data and control lines are connected to the appropriate edge connector pins. All other bus connections are brought out to a 16-pin socket pad. +5 volt regulator and all Molex connectors are provided; regulated +5 and ground are bused among the locations for up to 35 14-pin ICs. $24.95

THE MICRO WORKS P.O. BOX 1110, DEL MAR, CA 92014 714-942-2400

6809 RECORD MANAGEMENT SYSTEM RMS

DATABASE MANAGEMENT

- USER DEFINED DATA DICTIONARY AND RECORD FORMAT
- SCREEN ORIENTED, FORM FILL OUT TYPE OF ACCESS
- OPTIONAL TWO LEVEL RECORD HIERARCHY
- ALL FILES IN ASCII TEXT FORMAT, BASIC COMPATIBLE
- DIRECT ACCESS BY KEY FIELD, MULTIPLE INDEX FILES
- EXTENSIVE DOCUMENTATION, SAMPLE APPLICATION INCLUDED
- VERSATILE, PROFESSIONAL QUALITY REPORT WRITER
- BUILT-IN SORT/MERGE
- EASY TO USE



RMS is a complete DATABASE MANAGEMENT system for the 6809 computer. It runs under Flex, and supports the CT-82, or other CRT terminal. RMS is a set of five machine language programs that make up the most powerful business programming tool available for the 6809. It can be used by the relative novice, to implement an incredible variety of information storage and retrieval applications, without any programming. However, the programmer can use RMS as part of the solution to a larger problem, saving many hours of unnecessary program development time. RMS can be used to handle data input, editing, validation, on-line retrieval, sorting and printed reports. Custom data manipulation can be filled in by the user's BASIC programs.

WASHINGTON COMPUTER SERVICES
3028 SILVERN LANE
BELLINGHAM, WA 98225
A DIVISION OF **MICROPI**
1-206-734-8248

$200
SINGLE CPU LICENSE
5 OR 8 INCH DISK
TERMS: CASH/VISA/MC

DATA BASE MANAGEMENT SYSTEM

The **Universal Data Research Data Base Management System** provides the user and programmer, with a highly structured and efficient method of programming and data file handling.

The System consists of a Menu of programs to **Create, Build, Sort and Maintain Data Files, Generate Reports and Build User Programs.** All programs are written in **TSC Extended Basic.**

Included are Source Programs which contain all the common subroutines necessary for reliable, efficient programming for most applications.

UDRI also has a complete line of Data Base compatible end user programs such as:

- Order Entry/Accounts Receivable/Journal
- Purchase Order/Accounts Payable/History file
- Inventory Control/Bill of material
- Payroll
- Cash Disbursement/Cash Receipts
- Customer and Vendor Programs
- Numerous vertical market programs

The Data Base Management System allows the user to quickly modify and update the standard systems programs to fit Custom applications.

Introductory Price: $350.00

NY residents add sales tax

UNIVERSAL DATA RESEARCH, INC.
2457 Wehrle Dr.
Buffalo, N.Y. 14221
716-631-3011

SAVE

DISK

5" Soft Sector $2.77 each
5" 10-16 Sector $2.77 each

8" Single Side, Double Density $3.75 each
8" Double Side, Double Density $4.75 each

Minimum Order 10 (1 box)
Add $1.52 for 5" Plastic Box
Add $5.88 for 8" Plastic Box

Master Card-Visa-C.O.D. Accepted

Add $1.50 for Shipping & Handling

Dealer and Volume Discounts Available

SOUTH EAST MEDIA SUPPLY
P.O. Box 794
Hixson, TN 37343
615-870-1993

WAVE MATE INC.

Series 2000
Brings it all Together!

Hardware Features
- 2 MHz. 68B00 MPU
- Double Floppy Disk Drive - 368K bytes formatted
- 32K, 48K, or 64K byte dynamic RAM
- Intelli ent Video Terminal
- Commercial type riter keyboard with fu ctio keys and numberic pads
- 2 RS-232C serial ports

Software Features
- UCSD Pascal™ System Soft are Package
- 6800 Multi-tasking System (MTS6800)
- Business BASIC Compiler
- WORDMATE™ Word Processor
- Various Application Pac ages

™ UCSD Pascal is a trademark of the Regents of the University of California

Packaging
- Attractive, Compact, desk-top e closure
- Light-weight, highly portable
- Provision for 3 I/O Expansion modules
- Highly reliable, ease of maintena ce

Price: • Quantity 1 (one) end user price **$2,995** • **Attractive OEM/Dealer Discounts Available**

WAVE MATE INC.
18005 Adria Maru Lane
Carson, California 90746
213-532-4532
Telex 194369

EUROPEAN HEADQUARTERS
WAVE MATE INTERNATIONAL
159 Ch de Vleurget
1050 Bruxelles, Belgium
(02) 649-1070 Telex 24050

HEMENWAY ASSOCIATES
SOFTWARE SOURCE BOOKS™

Your 6800 is up and running with HEMENWAY ASSOCIATES' complete software system. Software Source Books provide a powerful yet extensible programming package for business, scientific, or personal uses.

Combining detailed descriptions with COMPLETE SOURCE CODE LISTINGS, these books explain the internal operations and algorithms used in HEMENWAY ASSOCIATES' popular systems software.

Imagine getting a complete 6800 software library, and at these suprisingly low prices.

Remember, these are not just books; they are Software Source Books™ complete software resources! Order them today; VISA and MASTERCHARGE accepted.

CP/68 OPERATING SYSTEM

The most powerful operating system available for the 6800 family of microprocessors, this disk-based system features great flexibility. The user can add commands for special purposes. A single transient Peripheral Interchange Program (PIP) transfers data between devices. The system is relocatable anywhere in memory and fits in less than 8K. Other features include device-independent I/O and dynamic file allocation.

U.S. $34.95
Int. $52.45

XA6809 MACRO LINKING CROSS-ASSEMBLER

This new two-pass program generates relocatable and linkable code (requires LINK68). Resident on any 6800 system, XA6809 lets you produce code for a 6809 right now. This assembler has full macro facilities and features a COMMON section for the production of ROMable code, Conditional Assembly and fast execution.

U.S. $24.95
Int. $31.50

LINK68 LINKING LOADER

This is a one-pass linking loader which allows separately translated relocatable object modules to be loaded and linked together to form a single executable load module, and to relocate modules in memory. It produces a load map and a load module in Motorola MIKBUG loader format. This book provides everything necessary for learning about this system and the nature of linking loader design in general.

U.S. $7.95
Int. $11.95

RA6800ML RELOCATABLE MACRO ASSEMBLER

This two-pass assembler produces a program listing, a sorted symbol table listing, and relocatable object code. The object code is loaded and linked with other assembled modules using LINK68. This book fully describes the 6800 assembly language and all major routines used, and includes flow charts, details on interfacing the assembler. Cross-referenced, showing all calling and called-by routines, pointers, flags and temporary variables.

U.S. $24.95
Int. $37.45

Structured BASIC language STRUBAL+™ COMPILER

The compiler features variable precision from 4 to 14 digits for business or scientific uses and Structured Programming forms. It produces Relocatable and linkable code. You can create data structures with mixed data types, COMMON and DUMMY sections. STRUBAL+™ includes a complete scientific package. It allows for string-handling and is extensible.

U.S. $49.95
Int. $74.95

Please send the following books:
_____ copies CP/68 OPERATING SYSTEM
_____ copies STRUBAL+™ COMPILER
_____ copies XA6809 CROSS ASSEMBLER
_____ copies RA6800ML MACRO ASSEMBLER
_____ copies LINK68 LINKING LOADER

HEMENWAY ASSOC., INC.
101 TREMONT STREET
BOSTON, MA 02108
(617)426-1931

FOR NORTH AMERICA, ADD $.75/BOOK POSTAGE & HANDLING or $1.50/BOOK FOR FIRST CLASS. ALL OTHER DESTINATIONS, ADD $2.00/BOOK POSTAGE & HANDLING or $3.50/BOOK FOR PRIORITY MAIL.

Name _____ Title _____ Company _____

Street _____ City _____ State/Country _____ Postal Code _____

☐ CHECK ENCLOSED IN THE AMOUNT OF $
☐ BILL VISA ☐ BILL MASTERCHARGE
CARD NUMBER _____ EXP. DATE _____
☐ FIRST CLASS/PRIORITY ☐ BOOK RATE

COMPUTER SYSTEMS CONSULTANTS INC.

1454 Lette Lane, Conyers, GA 30207
Telephone 404-483-1717 or 483-4570

SOFTWARE DEPARTMENT

680X DISASSEMBLER SYSTEM	$50.00
—runs under and processes any 680X instruction set	
—includes FLEX or SMOKE source programs	
—generates 6800/1 ASM code from 6800/1 program	
—generates 6809 ASM code from 680X program	
—optionally generates 6809 position-independent code	
—automatic lables, optional FCB, FCC, FDB, RMB	
—Disk-to-disk or console, commands from menu or disk	
—generates FLEX or SMOKE and user-defined names	
—alphabetic X-reference program for any 680X ASM code	
FULL-SCREEN FORMS DISPLAY FOR TSC X-BASIC	$25.00
—display and edit for terminals and video displays	
—requires 56K bytes on 6800 or 6809	
—interactive forms generator/documentor provided	
TSC BASIC RESEQUENCER PROGRAM	$25.00
—resequences TSC BASIC, X-BASIC, PC, XPC programs	
—partial and blank-resequence capabilities	
TSC X-BASIC DISK SORT/MERGE GENERATOR	$25.00
—interactively generates TSC XPC BASIC programs or subroutines for disk sort/merge	

(SPECIFY 5" or 8", FLEX OR SMOKE DISK FOR ALL PROGRAMS)

HARDWARE DEPARTMENT

I/O SELECTRIC INTERFACE BOARD	$25.00
—ASCII (serial or parallel) in 28-50V solenoids out	
—transparent serial interface (RS-232/TTL + CTS)	
SS-50 WIRE-WRAP BOARD (52-16 PIN EQUIVALENT)	$25.00
SS-30 WIRE-WRAP BOARD (32-16 PIN EQUIVALENT)	$12.50
SS-30 SERIAL INTERFACE BOARD (1 ACIA)	$10.00
SS-50 FRONT PANEL DISPLAY BOARD	$10.00
—16 LED's display first digit of address	
—great for debugging assembler programs	
25-WIRE RS-232 CABLE + CONNECTORS (8 ft.)	$20.00
34-WIRE 2-DRIVE DISK CABLE + CONNECTORS (8 ft.)	$30.00
50-WIRE 2-DRIVE DISK CABLE + CONNECTORS (8 ft.)	$39.00
STEEL CABINET (2-MINIFLOPPIES + POWER SUPPLY)	$40.00

—NO CREDIT CARDS — COD CHARGES EXTRA
POSTAGE/HANDLING 5% US 15% FOREIGN

F&D Associates
1210 Todd Road
New Plymouth, Ohio
45654

S-50 BUS

Tel. 614 - 592 - 5721

Send for free Catalog

Visa ~ Master Charge ~ C.O.D.

The TimeMinder

This month we introduce the TMB-1 "TimeMinder", a new clock/calender board for S50 systems. A 30 pin I/O card based on the OKI MSM5382, it keeps track of seconds, minutes, hours, day of week, day of month, month, year, and even allows for February 29th. Optional on-board battery backup and recharging circuit allows timekeeping to go on even when power is off.

The chip communicates thru an on board PIA. Five extra lines are brought out to a connector for user applications. One line is connected to a piezoelectric buzzer circuit for alarm applications. The board can generate interrupts to the system at 1024hz, 1hz, once per minute, and once per hour.

The board comes with assembly and checkout instructions and software assembly listing. A diskette is available separately for FLEX 2.0(tm)* systems. It contains the demonstration software, several FLEX utilities, and a patch for FLEX 2.0 that causes it to load the date into FLEX and display the date and time instead of prompting for the date. Source files are also included.

TMB-1	Bare Board and Documentation	$35.00
TMBFLX-1	5 1/4 diskette for FLEX 2.0	$18.00

Add $2.50 s/h to order. *FLEX is a trademark of TSC.

— POWERFUL INTEGRATED 6800, 6809 SOFTWARE —

All software is supplied in relocatable format and may be loaded anywhere in memory

6800 RELOCATING ASSEMBLER AND LINKING LOADER. The assembler supports relocatable and absolute code, labeled common blocks, 47 error messages, alphabetized or non-alphabetized cross reference table, 8 char global and local labels, TSC source compatability, and much more. The linking loader will link up any number of object modules and place the resultant object anywhere in memory that you wish.
M68-ASMB-1A-10, M68-LOAD-1A-10 .... $100.00

6800 ONE PASS LINK EDITOR. Functions in a similar manner to the linking loader except that the final object is stored back on disc as one object module. This module may then be loaded anywhere in memory by the linking loader (M68-LOAD-1A-10). This allows the creation of library of relocatable 'quick-load' modules.
M68-LNKA-1A-10 .... $45.00

6800 TWO PASS LINK EDITOR. Similar to the one pass link editor. Some optimization is performed.
M68-LNKB-1A-10 .... $45.00

6800 GLOBAL CROSS REFERENCE GENERATOR. Provides the capability to tell at a glance all object modules that reference a particular internal label.
M68-XREF-1A-10 .... $35.00

6800 OBJECT DISPLAYER. Lists the header of an object module. Parameters displayed are program name, internal labels, external labels, time, date, and identification.
M68-DISP-1A-10 .... $25.00

SPECIAL OFFER. All 6 M6800 programs above are available for **$200.00**

A 6800 users manual describing all 6 programs above is available for $15.00. This is refundable with your purchase of the assembler and linking loader. All 6800 software is supplied on 5-inch FLEX discs and runs with FLEX 2.0.

6809 RELOCATING ASSEMBLER AND LINKING LOADER. Supports features similar to those of the 6800 software above. Will also assemble 6800 and 6801 instructions.
M69-ASMB-1A-10 .... $100.00

The following 4 programs function in a similar manner to their 6800 counterparts.

6809 ONE PASS LINK EDITOR.
M69-LNKA-1A-10 .... $45.00
6809 TWO PASS LINK EDITOR
M69-LNKB-1A-10 .... $45.00
6869 GLOBAL CROSS REFERENCE GENERATOR.
M69-XREF-1A-10 .... $35.00
6809 OBJECT DISPLAYER.
M69-DISP-1A-10 .... $25.00

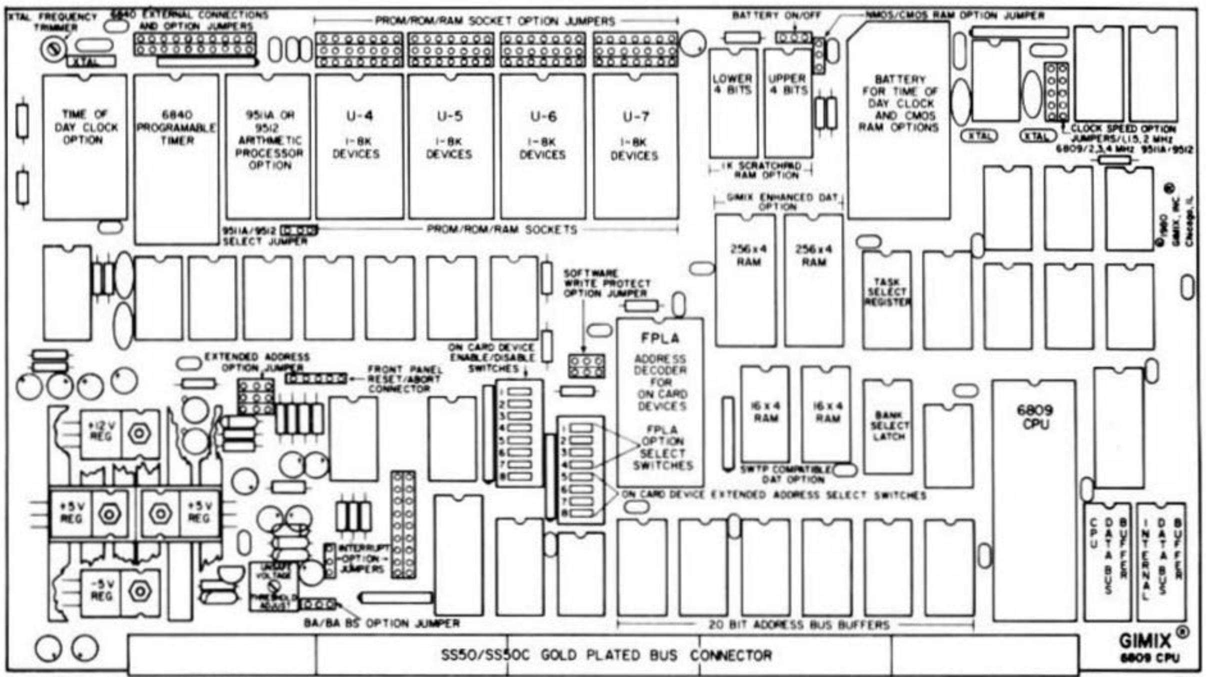
SPECIAL OFFER. All 6 M6809 programs above are available for **$200.00**

A 6809 users manual describing all 6 programs above is available for $15.00 (refundable with purchase of assembler and loader). All 6809 software is supplied on 5-inch FLEX discs and runs with FLEX 9.0.

A CATALOG of all of our Software Products is available FREE for the asking.

CINCITEK SOFTWARE
BOX 19365
CINCINNATI, OHIO 45219
(513) 751-6203

GIMIX PROUDLY PRESENTS!!!



GIMIX 6809+ FUNCTION LAYOUT

SS50C 6809 CPU

The 6809 CPU card will be available in a standard version and our 6809 PLUS version that is fully socketed to allow adding options at anytime.

+ A 6840 timer package that provides 3 independent 16 bit counters is included on all **6809 PLUS** cards.
+ A 9511 or 9512 Arithmetic Processors option with its own independent crystal that allows you to use 2, 3, or 4 MHz parts in any combination with the 6809 running at 1, 1.5, or 2 MHZ.
+ 1K of scratchpad RAM
+ A Time of Day Clock option with battery back-up. With this option you can also substitute 1K of CMOS RAM that will also be battery backed up.
+ User selectable processor speeds without having to change the crystal.

32K of PROM, ROM or RAM. Both versions have 4 sockets that can each hold from 1K to 8K parts. Single or multiple voltage parts can be used on the **PLUS** version. The standard version only allows the use of single voltage parts.

All on board devices and options can use extended addressing so that they will only respond to that page to which they are set.

The card is double buffered and allows versatility in the use of software and memory address control disciplines.

Please note that this card does **not** have an on board baud rate generator, and must be used in systems where baud rates (if needed) are provided elsewhere in the system.

GIMIX INC. 1337 WEST 37th PLACE • CHICAGO, IL 60609 • (312) 927-5510 • TWX 910-221-4055

'68' Micro Journal
3018 Hamlin Rd.
Hixson, TN 37343

Second Class Postage Paid
At Chattanooga, TN
ISSN 0194-5025